Desenvolupament d'una plataforma informàtica pel tractament, manipulació i visualització de dades cardíaques

Gerard Alcorlo Bofill

Índex de continguts

1 Introducció i Objectius

El diagnòstic per la imatge s'ha convertit en una eina de treball fonamental en la majoria de centres hospitalaris. A partir de dispositius de captació com poden ser la ressonància magnètica o la tomografia computeritzada, es poden obtenir representacions gràfiques (en forma d'imatges) dels òrgans interns d'un pacient. Amb aquesta informació els especialistes poden fer un diagnòstic molt més acurat ja que tenen informació del pacient que a simple vista no es pot obtenir.

El tractament i processament de les les dades que s'obtenen d'aquests dispositius de captació ha obert una nova àrea d'estudi i de treball en la qual s'han proposat diferents tècniques i algorismes. L'objectiu d'aquests algorismes es proporcionar mètodes i estratègies que facilitin la interpretació i la detecció de les patologies que estan representades en les imatges. En general, els algorismes que es proposen van lligats a una patologia concreta. O sigui que no és el mateix tractar patologies cardíaques, que problemes cerebrals o lesions òssies. La diferència està en el fet que el que li interessa detectar en una imatge a un cardiòleg, a un neuròleg i a un traumatòleg és molt diferent. D'altra banda les característiques de les imatges del cervell, del cor, o d'un ós tampoc són les mateixes (veure imatge Figura 1).

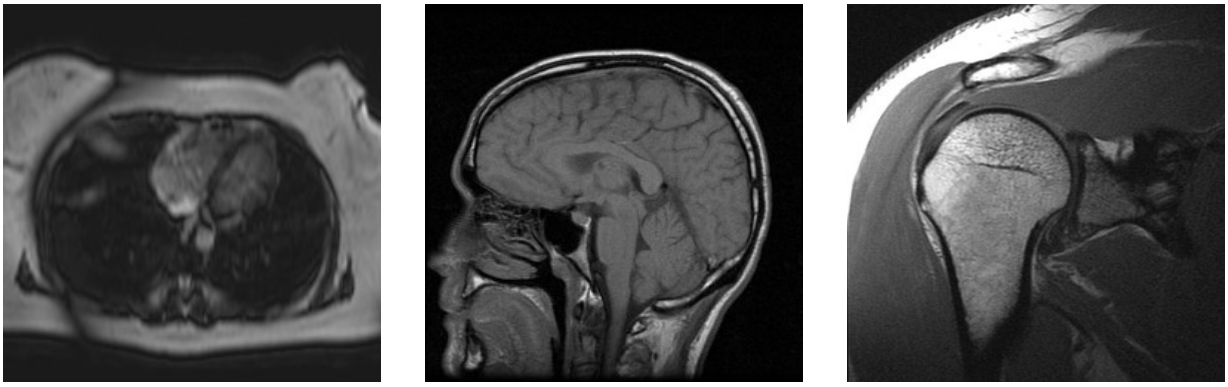


Figura 1: D'esquerre a dreta. MRI del cor. MRI del cap. MRI de l'espatlla

Com es pot observar a la Figura 1 les tres imatges, a part de referir-se a diferents parts del cos, tenen un aspecte molt diferent. La imatge del cor es veu més borrosa perquè el cor està en moviment i és difícil capturar-ne una instantània perfectament nítida. A més a

més, la imatge es correspon a una llesca d'una seqüència de imatges que permet reconstruir un model 3D. Les limitacions tècniques de la tecnologia actual impedeixen poder realitzar imatges de més resolució ja que llavors el soroll de les imatges esdevé molt important.

La imatge del cap també es correspon a una llesca extreta d'una seqüència de imatges. És més nítida que la del cor bàsicament perquè no es troba en moviment, exceptuant la circulació de la sang dins el cervell. I finalment la ressonància de l'espatlla com que es tracta d'una zona on no hi ha moviment, s'ha pogut realitzar una única captura a més resolució obtenint així una imatge de millor qualitat.

Podem afirmar que el processament de dades cardíaques és, sinó el que més, un dels més complexes de tractar. El problema principal és que a diferència d'altres parts de l'organisme, el cor del pacient està en moviment continu. Aquest moviment queda representat en les imatges generades pels aparells de captació en forma de soroll. Aquest soroll no només dificulta la detecció de les patologies per part dels cardiòlegs i els especialistes sinó que també en moltes ocasions limita l'aplicació de certes tècniques i mètodes. Així per exemple, l'aplicació de mètodes de visualització 3D (mètodes que permeten generar una representació 3D d'un òrgan) que poden aplicar-se fàcilment en visualització de dades del cervell no són aplicables sobre dades de cor.

En aquest projecte ens centrarem en el processament, tractament i visualització de dades cardíaques.

1.1 Antecedents

El Grup d'Informàtica Gràfica de la Universitat de Girona, juntament amb l'Institut de Diagnòstic per la Imatge (IDI) de l'hospital Dr. Josep Trueta, està col·laborant en el desenvolupament de noves eines informàtiques que donin suport al diagnòstic. Una de les prioritats actuals de l'IDI és el tractament de malalties cardíaques. Actualment les eines informàtiques de les que es disposen per poder tractar aquest tipus de patologies resulta insuficient. Tanmateix, es disposa d'una plataforma anomenada Starviewer que integra les operacions bàsiques de manipulació i visualització de dades mèdiques.

1.2 Objectius

L'objectiu d'aquest projecte és el de desenvolupar i integrar en la plataforma Starviewer els mòduls necessaris per poder tractar, manipular i visualitzar dades cardíaques.
Per assolir aquest objectiu caldrà estudiar, dissenyar i implementar els mòduls que permetin:

- Visualitzacions 2D, és a dir interpretar les dades que s'obtenen a través de l'aparell de captació i generar representacions 2D. El protocol aplicat per guardar les dades de cor és específic d'aquest tipus de dades, per tant s'haurà d'estudiar i interpretar quin és aquest protocol.
- Visualitzacions 3D. Normalment la informació que s'adquireix del pacient és només d'una llesca del cor (visualització 2D). En el cas de tenir informació de diferents talls del cor es podria construir un model 3D i fer-ne la posterior visualització. L'adquisició d'un model 3D segueix un altre protocol diferent. Caldrà per tant estudiar quin és aquest protocol i com podem generar i visualitzar el model 3D.
- Visualitzacions 4D per realitzar estudis funcionals. Per visualitzacions 4D entenem visualitzacions en les que s'incorpora el temps, és a dir poder simular el batec que fa el cor. Una vegada aconseguides les visualitzacions 2D i 3D s'intentarà representar el temps en les imatges. Una forma de representar el 4D és generant seqüències de vídeo que permetin simular el batec del cor.
- Exportar les dades a diferents formats del que proporciona el dispositiu de captació permetent, per exemple la generació de fitxers de vídeo.
- Visualització i manipulació per zones d'interès, procés que també es coneix amb el nom de selecció de ROIs (Region of Interest).
- Interrogació de models que inclou la capacitat de mesurar distàncies, àrees, volums, angles sobre la imatge, etc.

La part final del projecte consistirà en la validació en casos reals de tot el codi desenvolupat. Caldrà per tant la coordinació amb els tècnics radiòlegs de l'hospital Josep Trueta.

El fet que el projecte s'hagi d'integrar en la plataforma Starviewer implica una part de treball en equip amb el grup de treball del Starviewer. Així doncs podem dir que un dels objectius serà també la meva incorporació i adaptació al grup de treball.

1.3 Pla de treball

Per assolir els objectius que ens hem marcat en el projecte el pla de treball que seguirem l'hem dividit en dues etapes.

- La primera etapa consistirà en un estudi de totes les eines necessàries per poder començar a implementar.
  - → D'una banda s'estudiaran els diferents passos que cal seguir per passar de les dades que s'obtenen dels pacient a les imatges que es veuen en pantalla. Aquest procés és el que es coneix com el procés de visualització.
  - → A continuació ens centrarem en el cas de les dades cardíaques, és a dir, què cal fer per aplicar el procés de visualització sobre dades de cor. Hauran d'estudiar-se els diferents protocols de captació, que com hem dit abans varien en funció del tipus d'exploració que es realitza sobre el pacient.
  - → S'estudiarà també l'estructura de la plataforma Satrviewer per determinar els mòduls que caldrà ampliar i/o modificar per assolir els nostres objectius.
  - → S'estudiaran les eines de desenvolupament fixades pel grup de treball del Starviewer.

Amb aquesta primera fase d'estudi considerem que tindrem el coneixements necessaris per poder assolir un a un els diferents objectius que ens hem marcat.

- La segona etapa consistirà en el disseny i implementació dels diferents mòduls. Per cada objectiu caldrà aprofundir en els coneixements que siguin necessaris per poder-lo assolir.

En quan al temps la planificació que hem fet per cadascuna de les etapes és la que es mostra en el calendari següent.

Febrer ***2006***

Dl	Dt	Dc	Dj	Dv	Ds	Dg
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28					

Març ***2006***

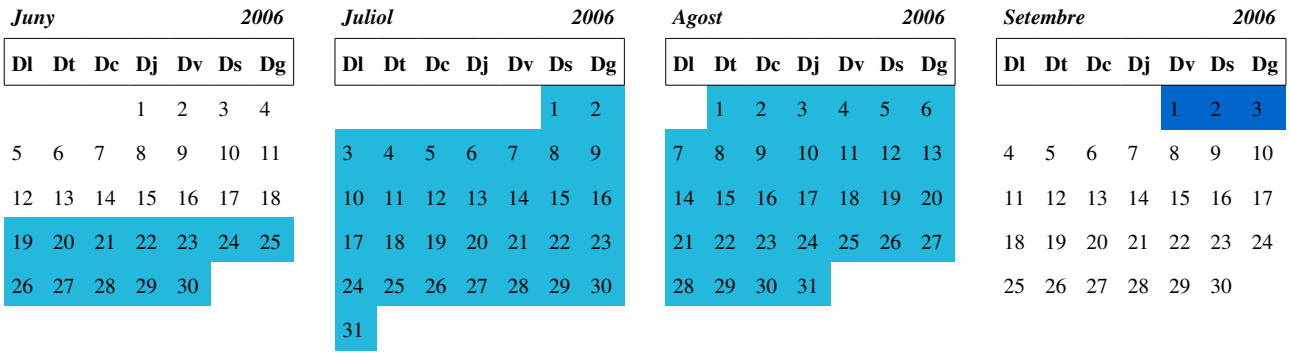
Dl	Dt	Dc	Dj	Dv	Ds	Dg
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Abril ***2006***

Dl	Dt	Dc	Dj	Dv	Ds	Dg
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Maig ***2006***

Dl	Dt	Dc	Dj	Dv	Ds	Dg
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				



- Assignació del projecte
- Reunió amb l'equip de desenvolupament i coordinació del Starviewer
- Instal·lació de Linux, Starviewer i compilació de llibreries
- Familiarització i aprenentatge de VTK
- Aprenentatge de QT
- Familiarització i aprenentatge de ITK
- Estudi del Starviewer
- Disseny, implementació, proves i documentació
- Formateig i últims retocs de la documentació

1.4 Metodologia

Aquest projecte desenvolupat sobre el paradigma de la orientació a objectes s'ha realitzat seguint la metodologia de l'*Extreme Programming* (XP). Per ser sincers, no és que a l'inici del projecte ens proposéssim seguir pas a pas aquesta metodologia. Però en part obligats pel temps que disposàvem es va decidir seguir les bases d'aquesta metodologia i a l'hora de fer els dissenys i diagrames utilitzar el llenguatge UML (*Unified Modeling Language*[UML]) que ja ens era familiar gràcies a les assignatures cursades al llarg de la carrera.

L'avantatge d'utilitzar *Extreme Programming* és que vas fent seguir les diferents vessants del projecte alhora. El disseny, la implementació i les proves esdevenen un procés iteratiu i simultani, fet que provoca que a mida que el codi va evolucionant es pensin i refacin els diagrames de classes.

També s'ha seguit la metodologia XP pel que fa a la relació amb el client, en aquest cas el personal mèdic encarregat d'utilitzar l'aplicació. De manera, més o menys periòdica es mirava d'ensenyar al tècnic especialista en fer anar la màquina de ressonàncies, la feina realitzada per tal d'aclarir quines opcions podien interessar més els metges.

1.5 Estructura del Document

La memòria que presentem l'hem estructurat en 14 capítols. En el primer capítol es situa el projecte i es marquen els objectius que es volen assumir tot especificant quina serà la metodologia de treball utilitzada per fer-ho. El segon capítol explica tot el que fa referència al procés de visualització de dades. Des de que són captades fins que són mostrades a la pantalla de l'ordinador. El tercer capítol explica quin és el protocol que es segueix en la fase de captació de dades i quin paper tenen les diferents etapes del procés de visualització quan es treballa amb dades de cor provinents de ressonàncies magnètiques. En el quart capítol es veu com es guarden les dades a disc, es fan cinc cèntims del format d'imatge utilitzat en l'àmbit mèdic i es mostren els principals tipus d'imatges de ressonàncies de cor amb què es treballa. En el capítol 5 es comenten les eines utilitzades per desenvolupar l'aplicació, el llenguatge de programació utilitzat i els coneixements que es tenien al respecte. Finalment s'inclou un esquema comentat de la versió del Starviewer a partir de la qual s'inicia aquest projecte. El capítol 6 mostra totes les funcionalitats que disposarà l'aplicació agrupades per objectius i mostrant els casos d'ús amb les fitxes que en detallen les explicacions corresponents. Al setè capítol es parla del visor 2D que permet la visualització de imatges en dues dimensions així com la seva reproducció. També permet exportar el vídeo a format mpeg2, però aquesta part es veu en detall al capítol 8. El capítol 9 parla de la visualització de imatges 3D i 4D per fer-ho ens presenta el visor 3D-4D per plans. El capítol 10 mostra els canvis necessaris per permetre la visualització de models 3D a partir de les classes existents al Starviewer inicial. Al capítol 11 es proposen una sèrie d'ampliacions i millores de l'aplicació. Al 12 s'exposen les conclusions extretes un cop acabat aquest projecte. I finalment al capítol 13 s'explica el funcionament de l'aplicació amb un manual d'usuari il·lustrat.

Al final de tot abans de la bibliografia consultada s'inclouen dos annexos. Un mostra tots els camps presents en un fitxer DICOM provinent d'una ressonància magnètica del cor i a l'altre es fa una breu explicació de la compressió utilitzada en el format mpeg2.

2 Procés de visualització

El procés de visualització és el procés que cal aplicar per passar de les dades que s'han obtingut del pacient a la imatge en pantalla. Aquest procés està format per les quatre etapes que es descriuen a continuació i que es mostren a la Figura 2.

Figura 2: Etapes del procés de visualització

2.1 Captació de dades

La primera etapa del procés de visualització és la captació de dades. L'objectiu principal és l'obtenció del conjunt de dades que cal interpretar. Aquestes dades s'obtenen mitjançant simulacions o bé a través de dispositius especialitzats de captació.

Entre aquests dispositius de captació trobem entre altres: La tomografia Computeritzada (CT), utilitzada principalment per la detecció de sòlids d'alta densitat, com ara els óssos. La ressonància magnètica (MRI) usada especialment per l'estudi de teixits. La tomografia per emissió de protons (PET) utilitzada per detectar la dispersió de fluids, i molt apropiada per l'estudi de funcions metabòliques.

Independentment de les característiques de les dades capturades per tots aquests aparells, tots ells tenen en comú que retornen les imatges i les guarden en el format DICOM. A més a més, una altre de les característiques que tenen en comú el tipus de dades amb el qual treballem, és que sempre segueixen una distribució espacial regular i que sempre ens venen distribuïdes sobre plans.

Generalment en el moment de fer les adquisicions de dades es consideren trens direccions de captació. Cada direcció es paral·lela a un dels eixos de coordenades. Aquestes direccions es coneixen amb els noms de axial, coronal i sagital. A la Figura 3 es pot veure una imatge capturada al pla axial, una al sagital i una al pla coronal.

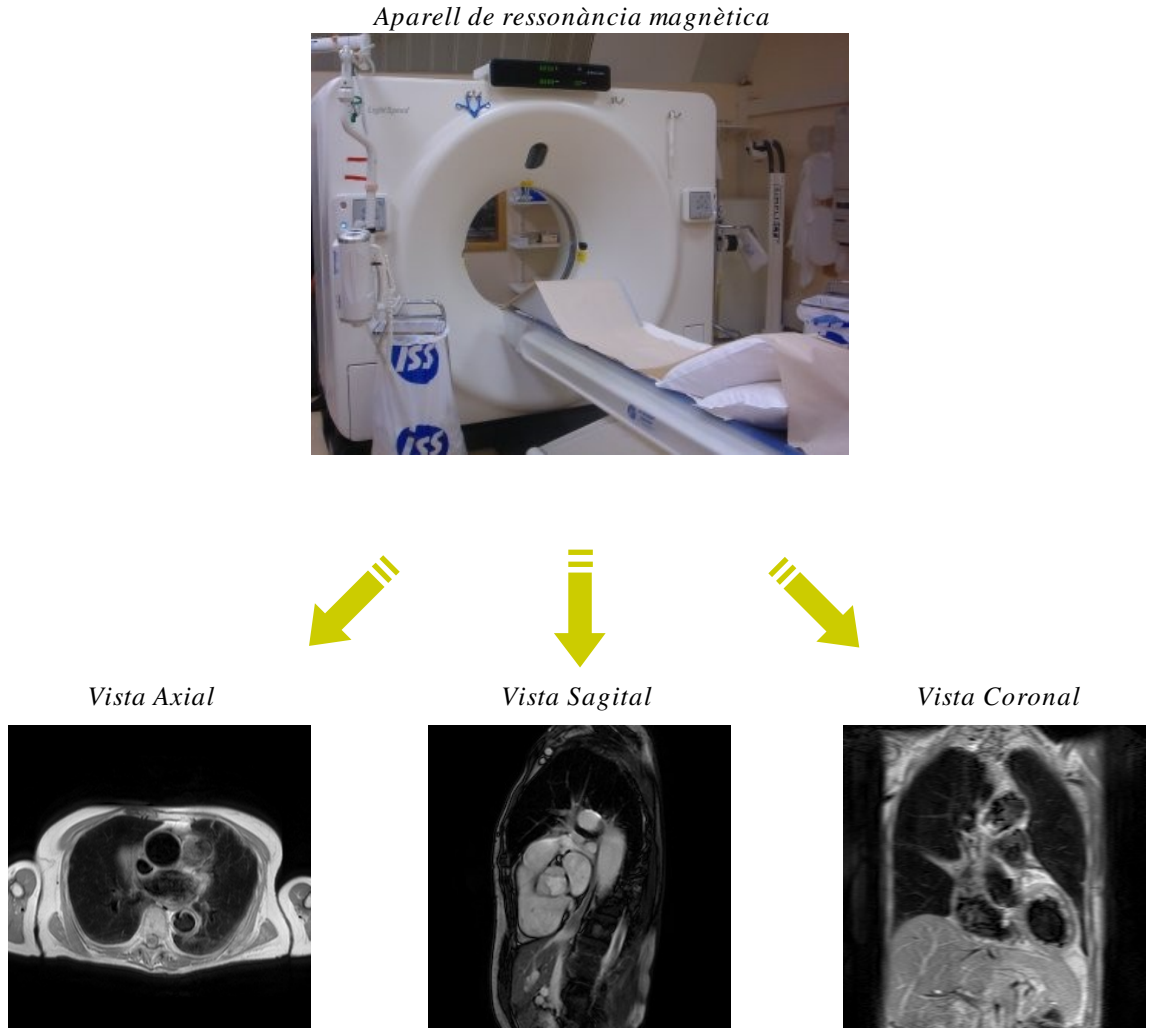


Figura 3: Captures provinents del pla axial, sagital i coronal

2.2 Definició del model de representació

En aquesta etapa, es defineix el model de representació que s'usarà per representar les dades obtingudes en l'etapa anterior. Hem de tenir en compte que la sortida del procés de captació és un fitxer en format DICOM en el que tenim una o més imatges que representen dades d'alguna part de l'organisme del pacient. Podem pensar que és com si haguéssim simulat el tall en seccions planars d'un òrgan. Apilant aquestes llesques podríem reconstruir en 3D l'òrgan en qüestió, obtenint així un model volumètric.

El model més usat en el camp de la medicina per representar les dades és el model de voxels proposat per Kaufman. Aquest model subdivideix l'espai en un conjunt de cubs o paral·lelepípedes de les mateixes dimensions, seguint una malla regular. Cadascun d'aquests cubs, o paral·lelepípedes, rep el nom de voxel. Sobre cada voxel s'hi representaran les dades que s'han obtingut del pacient. A l'esquema de la Figura 4 podem veure una representació gràfica de com passem del conjunt de llesques representades en el fitxer DICOM al model de voxels. Partirem de les dades que s'han obtingut del dispositiu de captació, guardades en el fitxer DICOM. Els diferents plans o talls s'apilaran un sobre l'altre intentant reconstruir l'òrgan examinat en 3 dimensions (3D). En el model de voxels el que s'ha fet és donar volum als píxels de manera que un píxel sigui vist com un petitíssim cub en 3 dimensions. El píxel i el voxel guardaran la mateixa informació, el valor de intensitat del píxel / voxel en aquell punt.

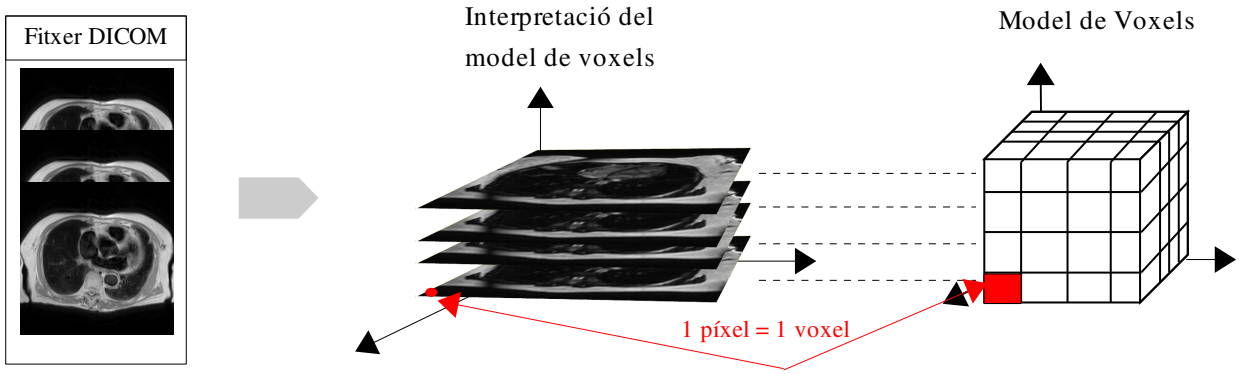


Figura 4: Interpretació del model de voxels proposat per

2.3 Selecció de primitives

Una vegada definit el model de voxels, la següent etapa en el procés de visualització és la selecció de les primitives geomètriques que s'utilitzaran per generar la imatge final.

Els algorismes de visualització variaran en funció de si es treballa sobre el model de voxels inicial o si es treballa sobre un model simplificat, entenem per model simplificat un model que es pot obtenir a partir del model de voxels. En el primer cas parlarem de visualització directa del volum. En el segon cas parlarem de visualització de models reconstruïts.

Per obtenir un model reconstruït a partir del model de voxels podem aplicar les tres estratègies que es descriuen a continuació.

2.3.1 Reconstrucció 0D

La unitat de treball seran els voxels que formen el model. El que es farà serà treballar amb un nombre reduït de voxels, que no serà res més que un model simplificat de l'original. El fet de descartar informació pot ser interessant si el que volem és que les visualitzacions siguin més ràpides. A la Figura 5 es veuen de color vermell els punts escollits del model, així com la correspondència amb les imatges originals.

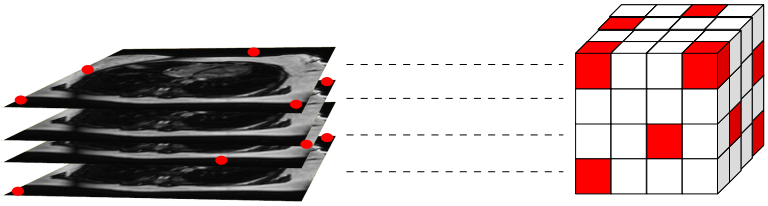


Figura 5: Equiparació entre els punts seleccionats utilitzant reconstrucció 0D i el model

2.3.2 Reconstrucció 1D

De la mateixa manera que abans s'ha dit que la unitat bàsica era el voxel, ara direm que és el pla. Serà a partir de les llesques capturades que n'extraurem un contorn. Extreure un contorn voldrà dir buscar els voxels que tenen un valor determinat. Aquest procés es repetirà per cada llesca i la imatge final s'obté a partir de la visualització de tots els contorns. Hi ha variants de l'algorisme que apart d'extreure els contorns apliquen un procés d'unió entre els contorns de cada llesca. Com es pot desprendre de l'explicació, en aquest cas també es treballa amb un model simplificat. Vegis la Figura 6 a on a cada llesca se n'ha escollit una franja de punts la visualització del qual ens dibuixaria una mena de cilindre que en comptes de tenir parets tindria cercles.

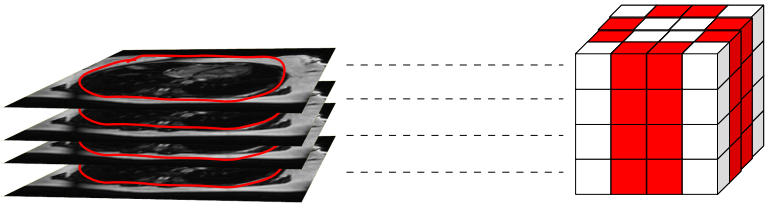


Figura 6: Equiparació entre els punts seleccionats utilitzant reconstrucció 1D i el model

2.3.3 Reconstrucció 2D

En aquest tipus de reconstrucció es treballa amb tots els valors del model de voxels. Consisteix en la reconstrucció de superfícies i juntament amb la visualització directa de volums és la més aplicada.

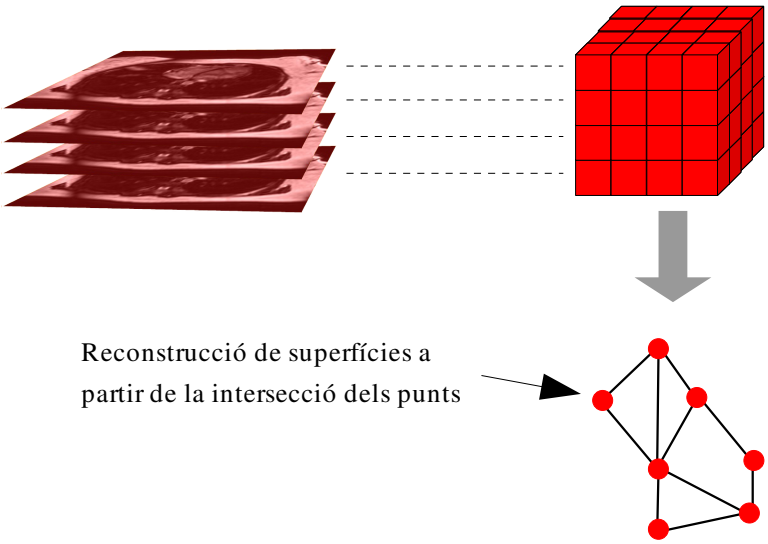


Figura 7: Equiparació entre els punts seleccionats utilitzant reconstrucció 2D i el model

2.4 Visualització

La darrera etapa és la de visualització. En aquesta fase s'aplica alguna tècnica de visualització sobre el model de dades (o el model simplificat) per tal d'obtenir la imatge final. Aquest mètode de visualització variarà en funció del model de representació.

La tècnica de visualització haurà de determinar quins atributs gràfics s'assignen als diferents valors de propietat representats en el model per tal que es pugui obtenir una representació gràfica del mateix. Evidentment el tipus d'atributs i les tècniques a aplicar variaran en funció del tipus d'informació que volem visualitzar i del tipus d'informació que tinguem representada en el model. No és el mateix visualitzar el model de voxels que visualitzar un model reconstruït.

Tot i que el que més preocupa a l'usuari és la visualització, sovint també vol interrogar el model, amb l'objectiu de calcular distàncies, volums, etc. Aquests càlculs li seran útils per tal d'interpretar i treure conclusions sobre el model visualitzat.

3 Processament de dades cardíaques

En aquest capítol ens centrarem en el procés de visualització explicat en el capítol anterior però donant importància a les característiques a tenir en compte en el cas que les dades a processar siguin dades de cor.

3.1 Captació de dades

En els pacients que pateixen algun tipus de patologia cardíaca se'ls hi poden fer diferents tipus d'exploracions. En aquests projecte en centrarem en les exploracions realitzades a partir d'aparells de ressonància magnètica. Per tant les imatges que tractarem en aquest projecte s'han obtingut a partir de ressonàncies magnètiques.

La ressonància magnètica és una tècnica que permet obtenir imatges del cos a partir d'un escàner que es caracteritza per utilitzar un imant i ones de ràdio. Mitjançant unes antenes especials l'escàner MRI rep les ones de ràdio provinents del cos i processa el senyal a l'ordinador per tal de generar imatges molt clares de l'anatomia. La fiabilitat de les dades obtingudes dependrà de la precisió de l'aparell utilitzat en la captació de les mateixes.

És una tècnica no dolorosa i no perjudicial per la salut que és especialment útil a l'hora d'estudiar l'estructura i el funcionament del cor i els vasos sanguinis. Permet la visualització de informació que no es pot veure amb altres tècniques com puguin ser els raig X o ultrasons.

3.2 Protocol d'exploració en les patologies cardíaques

Generalment, als pacients afectats d'algun tipus de patologia cardíaca quan se'ls hi fa una exploració usant un aparell de ressonància magnètica, aquesta és d'un dels tres tipus que explicarem a continuació. Cal dir que aquestes exploracions segueixen un protocol estàndard que és el que es segueixen a la majoria d'hospitals.

- **Black Blood**, en aquest tipus d'exploració el pacient ha d'aguantar la respiració i es capturen imatges 2D en diferents plans de l'espai. El tipus de imatges que obtenim és el mostrat a la Figura 8, i serveixen per fer una exploració inicial per tal de ubicar el cor les artèries, etc. Se'n capturen poques imatges, 3 o 4.

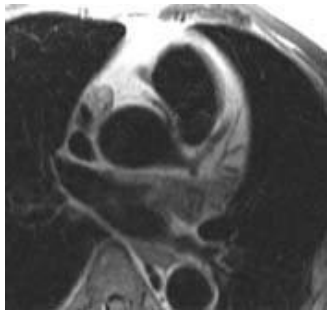


Figura 8: Imatge del tipus black blood

- **Balanced (FFE / TFE)**, en aquest tipus d'exploració es capturen imatges 2D del cor, i es parla de l'eix llarg i de l'eix curt del cor. Si tenim en compte la forma del cor ràpidament podem interpretar quin és l'eix llarg i quin és el curt. Quan es fa una prova sobre el l'eix llarg el que es sol fer és agafar una sola llesca que travessi longitudinalment el cor. Una prova típica sobra l'eix llarg és la quatre càmeres (4CH), dita així perquè es veuen clarament les quatre cavitats del cor. En canvi quan es parla de l'eix curt la prova consisteix en agafar tres llesques perpendiculars a l'eix llarg. En les imatges de les Figures 9 i 10 es pot veure exactament la direcció de les seccions que s'agafen en el cor quan es captura sobre l'eix curt i sobre el llarg.

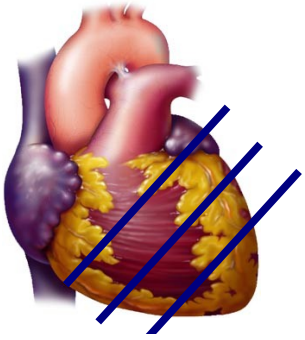


Figura 9: Captures sobre l'eix curt del cor

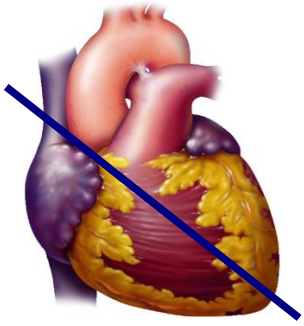


Figura 10: Captura sobre l'eix llarg del cor

En aquesta prova es capturen imatges que són útils quan són visualitzades com una seqüència de vídeo. Cal tenir present que la captura pot durar uns trenta segons, però el que tindrem a les imatges sempre serà un únic batec, que si reproduïm cíclicament dóna la sensació que el cor estigui bategant contínuament. La captura d'aquestes imatges és força complicada perquè les imatges només es capturen en el punt en què la respiració i el batec del cor estan sincronitzats de una forma determinada.

- **Perfusió / Viabilitat.** Aquestes són les proves típiques per tractar el que coneixem com a infart de cor (agut de miocardi). S'injecta un líquid a la sang del pacient que al ser analitzada a través de la ressonància magnètica destaca per sobre de la resta de teixits. S'utilitza per veure exactament quines són les parts del cor o venes del voltant que reben sang (Figura 11). Als metges tant els interessa veure com la sang circula dins el cor, com veure fins on arriba aquesta.

  Amb la perfusió el que es veu és quines són les zones que reben sang de bones a primeres i amb la viabilitat es veuen les zones que passada una estona també aconsegueix arribar-hi la sang. Els dos tipus de zones són d'interès per als radiòlegs.

Figura 11: Imatge capturada durant una prova de perfusió / viabilitat

Un cop feta la primera prova de Black Blood la presència d'un radiòleg és necessària perquè serà ell qui decidirà quines proves més són interessants per tal de diagnosticar. Amb això es vol dir que tot i haver-hi un protocol no totes les proves tindran el mateix número de seqüències, sinó que el nombre variarà en funció de si el metge necessita diferents vistes del cor o creu que hi ha proves que s'han de repetir. Per aquest motiu un estudi contindrà un nombre variables de seqüències que dependrà del que vulgui veure-hi el metge.

Cal destacar que les proves duren entre una hora i una hora i mitja. Part d'aquest temps es deu al fet que hi ha proves que entra una i altre ha de passar un quartet d'hora. És cert que aquest temps es pot aprofitar i s'aprofita per fer altres proves. Però no sempre pot ser així ja que sovint per decidir quina prova cal fer implica tenir els resultats de l'anterior.

És evident que tenir una hora una persona malalta del cor fent-li proves on se li demana que aguanti la respiració no és bo. A més es neguiteja i no sempre s'està tant quiet com caldria per la prova, de manera que s'han de repetir i encara és pitjor pel malalt. Amb tot això es vol fer entendre, que aquestes proves són útils i es fan servir, perquè és el que a dia d'avui permet la tecnologia. Però de cares al metge poder tenir el cor amb 3D i en moviment, conegut com a 4D seria perfecte perquè llavors ell n'agafaria les vistes que més

li convindrien i de cares al malalt també seria positiu perquè no s'haurien de fer tantes proves sinó que amb una de sola ho tindria tot fet i no s'hauria d'esperar que el metge decidís quines proves venen a continuació.

El problema que hi ha és que les captures que es fan avui en dia no són prou bones i encara no són útils als radiòlegs. **Ni el programari ni les màquines de ressonància magnètica que tenen la majoria d'hospitals posseeixen la tecnologia necessària per obtenir bons resultats. Com que només és qüestió de temps s'ha volgut que aquest projecte fes una petita incursió dins el camp del 4D.**

Per acabar aquesta secció recordem que independentment del tipus de prova que es realitzi la informació provinent de la màquina de ressonància sempre ens ve donada en forma de fitxer DICOM.

3.3 Definició del model de voxels

Per generar el model de voxels, a part de la mida de les imatges i el nombre, s'utilitza la distància entre els píxels, la distància entre les llesques i la orientació de les captures. El que es farà serà col·locar cada llesca una a sobre de l'altre amb l'ordre correcte. Conèixer la distància entre píxels i entre llesques ens permetrà donar les proporcions adequades al model.

Podem diferenciar dos tipus de models, els 3D i els 4D. Un model 3D estarà format per una pila de llesques que al passar el píxels a vòxels ens donarà la sensació de volum. El 4D va un pas més enllà i contindrà varis models 3D. Un 4D consisteix en una sèrie de volums (llesques que poden crear un volum) que formen un dinàmic. S'entén per dinàmic a les captures realitzades al cor durant un període de sístole i diàstole.

3.4 Selecció de primitives

En el nostre cas treballarem directament amb tot el model de voxels. Per tant aquesta etapa del procés de visualització no requereix cap tractament especial.

3.5 Visualització

La darrera etapa del procés és la generació de la imatge final. Aquesta etapa és la que es pretén resoldre en el projecte, ja que com hem dit els nostres objectius són obtenir visualitzacions 2D, 3D i 4D.

4 Codificació de les dades de cor

Quan es realitzen les proves a un malalt cardíac, es diu que se li fa un estudi. Aquest estudi constarà de vàries proves tal i com ja s'ha comentat anteriorment.

El que ens interessa de cares a aquest projecte és determinar la informació que ens retorna l'aparell de ressonància. La màquina està connectada a un ordinador, quan es fan proves a un malalt es diu que es fa un estudi i com a tal l'ordinador destina una carpeta a aquest estudi. A més a més, per cada prova que es realitzi apareixerà una subcarpeta a l'estudi (vegis Figura 12).

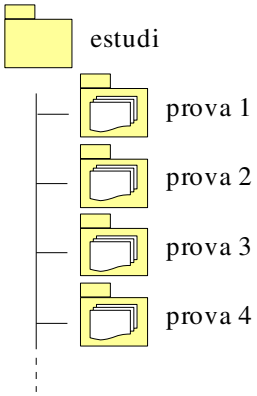


Figura 12: Fitxers resultants de fer una ressonància a un pacient

Dins de cada prova hi haurà un nombre variable de fitxers amb les imatges capturades. El format d'aquestes imatges és DICOM.

4.1 El format DICOM

El DICOM va néixer al 1983 com a fruit de l'esforç conjunt del Col·legi d'Estats Units de Radiologia (ACR) i l' Associació Nacional de Fabricants Elèctrics (NEMA). El seu objectiu era desenvolupar uns medis estàndards per usuaris d'imatges mèdiques digitals i per a la comunicació entre els dispositius i les màquines captadores. La missió d'aquest grup consistia en estandarditzar les comunicacions i la imatge digital, desenvolupant una interfície entre els equips radiogràfics i d'altres dispositiu que l'usuari desitgés connectar. A més de les especificacions per la connexió de hardware, l'estàndard incloïa un diccionari d'elements de dades necessaris per la interpretació i la visualització d'imatges. Finalment al 1985 en la reunió anual de la RSNA va publicar la primera versió de l'estàndard ACR-NEMA, el DICOM 1.0

Un arxiu DICOM pot contenir una o vàries imatges a més d'informació relacionada amb les imatges. Aquesta informació es guarda a la capçalera del fitxer i es divideix en quatre àrees, que són les següents :

- ***Pacient***: Aquesta àrea conté la informació relacionada amb el pacient, al qual pertany la imatge, dades com l'edat del pacient en el moment de capturar la imatge, nom, identificador de pacient, pes, altura, etc.. Tot tipus d'informació que pot ser rellevant per al metge, per ajudar a fer el diagnòstic.
- ***Estudi***: En aquesta àrea es guarda la informació referent a l'estudi d'un pacient en un determinat dia. Conté informació com l'hora i data de l'estudi, modalitat d'estudi. Un estudi pot contenir diferents sèries.
- ***Sèrie***: Una sèrie guarda la informació de cadascuna de les seqüències d'imatges realitzades en un mateix estudi. Conté informació com l'hora i data de la sèrie, modalitat de la sèrie, part del cos capturada, etc..
- ***Image***: Guarda els atributs referents a la informació de la imatge. Conté la informació necessària per poder visualitzar les imatges correctament com resolució de la imatge, posició, colors, etc..

DICOM s'utilitza per generar, guardar, mostrar, processar, enviar, recuperar, consultar i imprimir imatges mèdiques i els documents que se'n derivin, així com per gestionar-ne els flux de treball.

La capçalera dels fitxers serà d'utilitat en aquest projecte ja que gràcies a ella podrem diferenciar les imatges d'una mateixa prova que es corresponen a dues vistes diferents, cosa que també pot passar. Per veure quina és la informació que apareix a la capçalera d'una imatge de cor DICOM de ressonància magnètica, vegis l'annex A.

4.2 Contingut d'un estudi

Les proves dels estudis que provenen de ressonàncies podran contenir alguns dels tipus de imatges que es comenten a continuació.

- Poden contenir unes quantes imatges en vista coronal, és a dir, el pacient de cares. Aquestes imatges ens serveixen per ubicar exactament el cor els vasos, les venes, etc. (veure Figura 13)

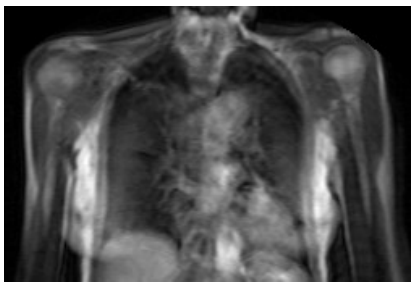


Figura 13: Vista coronal

- També podem trobar la captura d'una mateixa llesca en diferents instats de temps, captura que es coneix com a dinàmic del cor. En aquest cas veiem l'evolució de les diferents parts a mesura que el cor batega. (Veure Figura 14).

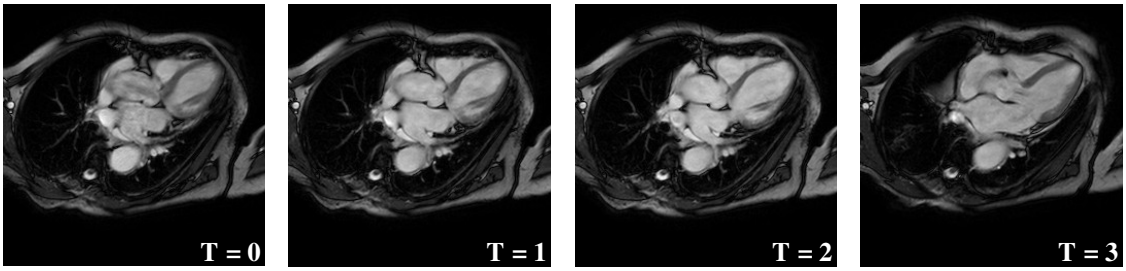


Figura 14: La mateixa llesca en diferents instants de temps

- Pot haver-hi un seguit de llesques en pla axial (Figura 15) que ens permeten reconstruir un model del cor. El més normal és que en aquest tipus de prova tinguem el model del cor capturat en diferents instants de temps, de manera que la seva reconstrucció podria permetre veure el cor en moviment amb 3D. Aquesta visualització es coneix com a 4D.

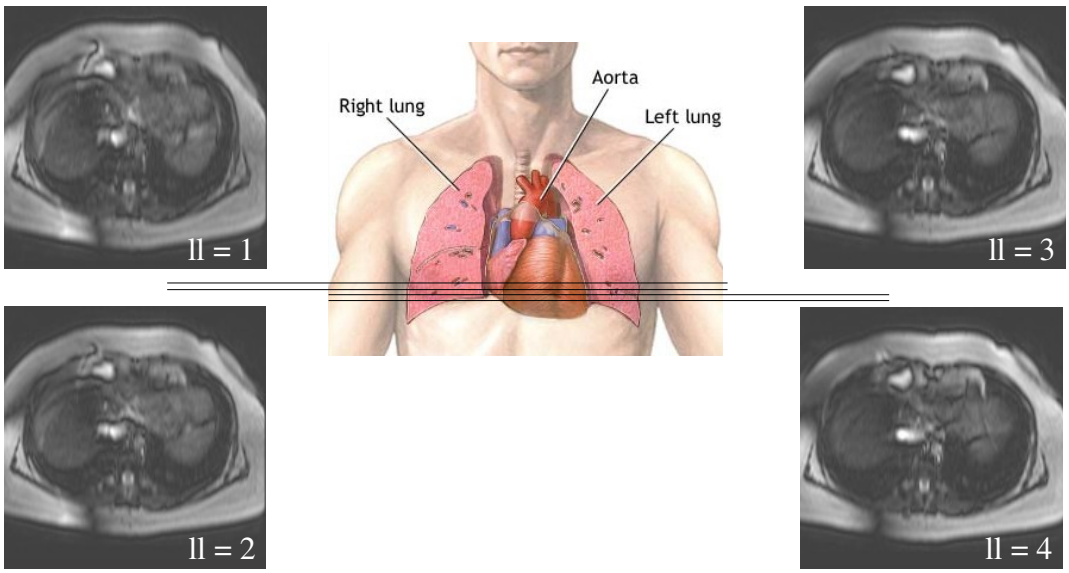


Figura 15: Cada llesca es correspon a un tall diferent del cor

- Finalment tindríem proves més específiques on les captures que es fan són específiques pel tipus de malaltia o problema del malalt. En el cas de que ja se sàpiga la patologia exacta es pot fer una exploració més restringida entrant paràmetres concrets a l'aparell de captació.

5 Entorn de Desenvolupament

Aquesta etapa tal i com es pot veure al calendari de planificació de l'apartat 1.3 té pràcticament la mateixa durada que el temps dedicat, a programar, desenvolupar i escriure documentació. La fase d'estudi previ és potser la dificultat més gran d'aquest projecte i és d'especial importància pel fet que el Starviewer es tracta d'un projecte complex amb un disseny avançat que al mateix temps ens ajuda i ens condiciona a l'hora de seguir afegint-hi funcionalitats. Sense un estudi previ i una comprensió del comportament i la interacció de les classes no es pot enriquir l'aplicació.
A part d'haver d'entendre l'estructura del Starviewer va ser necessari seguir alguns tutorials, llibres i exemples per saber utilitzar les 3 principals llibreries utilitzades en aquest projecte, QT, ITK i VTK.

El desenvolupament de l'aplicació està realitzat usant les llibreries ITK 2.4 (Insight Toolkit[ITK]), VTK 5.0 (Visualization Toolkit[VTK]) i Qt 4.0 sobre GNU/Linux. Les tres llibreries tenen la particularitat que són multiplataforma, de codi lliure, molt extenses, àmpliament utilitzades i es troben en continu desenvolupament.
L'entorn de programació serà el KDevelop i també utilitzarem el depuradors Gdb i Valgrind que venen integrats de manera gràfica dins el KDevelop. El llenguatge de programació, utilitzat és el C++ (Figura 16).

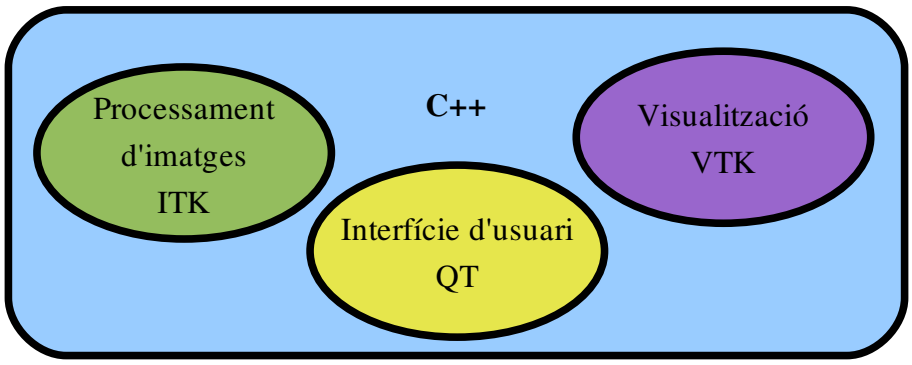


Figura 16: Llibreries i llenguatge utilitzat

La utilització d'aquestes llibreries ve condicionada pel fet que aquest projecte pretén ampliar i millorar el projecte Starviewer construït a partir d'aquestes. A continuació comentarem per a què serveix cada llibreria i perquè es van escollir

5.1 Llibreria QT

Les QT juntament amb l'aplicació QTDesigner permeten crear interfícies d'usuari agradables i visuals de forma ràpida, elegant i robusta. Permeten dissenyar finestres incorporant *widgets* de tota mena. Els *widgets* són els elements típics que podem trobar en un entorn visual com són botons, desplegables, menús, caixes de text, pestanyes, etiquetes, barres de desplaçament, etc. A part d'incorporar els *widgets* més típics també permeten crear-ne de personalitzats.

Tota la llibreria està construïda sota el paradigma de l'orientació a objectes, que resulta molt útil ja que l'aplicació Starviewer i aquest projecte en conseqüència segueixen el mateix paradigma i la integració i utilització de la llibreria és total permetent crear un codi net i entenedor. A més a més estan programades en C++ fet que en facilita la integració en el nostre projecte.

I finalment un altre característica a destacar de les QT és la utilització del patró *observer,* el qual han refinat i han convertit amb els anomenats *signals* i *slots.* És un sistema que permet a objectes de diferent tipus comunicar-se entre ells. Un *signal* és un avís que dóna un objecte quan ha passat algun succés (event), per exemple un botó pot emetre un signal "clicked" quan se'l prem. I un *slot* és aquella rutina que està esperant un senyal concret per a executar-se, per exemple finalitzar l'execució d'una aplicació. De fet, la implementació d'un *slot* és com la d'un mètode qualsevol. Per associar un *signal* amb un *slot* (o un altre *signal*) es crida la funció *connect*().

5.2 Llibreria VTK

Són unes llibreries pensades per al processat i visualització de imatges 3D. Com la resta de llibreries utilitzades són de codi lliure, multiplataforma i àmpliament utilitzades. Estan desenvolupades en C++ fet que en facilita la integració. A més a més, però, poden ser utilitzades des de llenguatges tant diferents com Python, Tcl o Java gràcies a un capa de programari extra que se'ls hi ha afegit.

Les VTK consten d'una sèrie d'objectes bàsics que combinats creen una escena. Aquests objectes bàsics són:

- Actor
- Light
- Camera
- Property
- Mapper
- Transform
- Renderer
- RenderWindow

Els *actors* representen aquells objectes que apareixen a l'escena, els *lights* il·luminen els actors en una escena i la *camera* defineix la posició de l'observador. O dit de manera més tècnica, la càmera és la que decideix des de quin punt i en quina direcció la geometria tridimensional es projecta sobre el pla que veurà l'espectador. Els actors estan definits per objectes de tipus *mapper*, *property* i *transform*. Un *mapper* representa la definició geomètrica de l'actor, un *property* representa els atributs de *rendering* de l'actor (com color, textura, especularitat, *shading*) i un objecte de tipus *transform* especifica la posició i orientació d'un actor, *camera* o *light*. Un *renderer* com el seu nom indica coordina el rendering de les llums, càmeres i objectes de l'escena. Finalment només ens queda el *renderWindow* que és la finestra que conté l'escena. Una mateixa escena pot contenir varis *renderers*.

Les VTK així com les ITK que es presenten a continuació es basen en el paradigma data-flow i els seus dos tipus bàsics d'objecte són el data-object i el process-object. A l'apartat 5.4 s'aclariran aquests dos conceptes.

5.3 Llibreria ITK

És un programari pensat per al processat d'imatges, que ofereix funcionalitats de segmentació i registre. El procés de segmentació consisteix en identificar i classificar determinades parts de les imatges, i el registre consisteix en combinar imatges provinents de diferents fonts per obtenir imatges que aportin més informació. En l'àmbit mèdic, el registre es pot fer per exemple amb les imatges extretes d'una ressonància magnètica (MRI) i les provinents tomografia computeritzada (CT).

Així com les altres llibreries, la ITK també està implementada en C++. Com les VTK també utilitza l'aplicació CMake per ser compilada de manera que pugui ser utilitzada en les diverses plataformes suportades. També permet ser utilitzada tant si es programa en Java, Tcl o Python.

La llibreria està dissenyada exprimint les possibilitats dels *templates* a la n-èssima potència. Això té el gran avantatge que els programes són molt eficients i que la majoria d'errors es poden detectar en temps de compilació. Però per contra fa que els errors de compilació siguin molt més críptics.

5.4 El paradigma data flow (VTK i ITK)

Una de les particularitats més importants que trobem presents tant a les ITK com a les VTK és el model orientat a objectes escollit. Si bé el model tradicional consisteix en encapsular en un mateix objecte el processament i l'emmagatzematge de les dades, hi ha una altre opció que consisteix en separar els objectes per, objectes de dades i objectes de processament, i és precisament aquesta opció que es coneix com a paradigma data-flow la que utilitzen tant les ITK com les VTK. De fet s'utilitza un model híbrid entre les dues aprofitant com sempre el millor de cada una ja que hi ha unes determinades operacions que al ser crítiques s'implementen dins els propis objectes de dades.

El fet de separar dades i algorismes dóna lloc a l'anomenat pipeline o xarxa de visualització, on hi intervenen els *data objects* i els *process objects*.

Els *data objects* representen la informació i ofereixen mètodes per crear-la, accedir-hi i esborrar-la. La modificació directa de les dades no és permesa i només ho poden fer els *process objects*. Els *data objects* es diferencien entre ells per la seva representació interna. En funció d'aquesta variarà la velocitat d'accés a les dades, l'eficiència d'emmagatzematge o la capacitat per interaccionar amb altres objectes.

Els *process objects* treballen a partir d'unes dades d'entrada per generar unes dades de sortida. Es classifiquen en "*source objects*", "*filter objects*", o "*mapper objects*", en funció de si inicien, mantenen o finalitzen el flux de dades. De manera genèrica als tres tipus de "*process objects*" se'ls anomena filtres. Tal i com es mostra a la Figura 17 el nombre d'entrades o sorties pot ser 0 o més d'una.

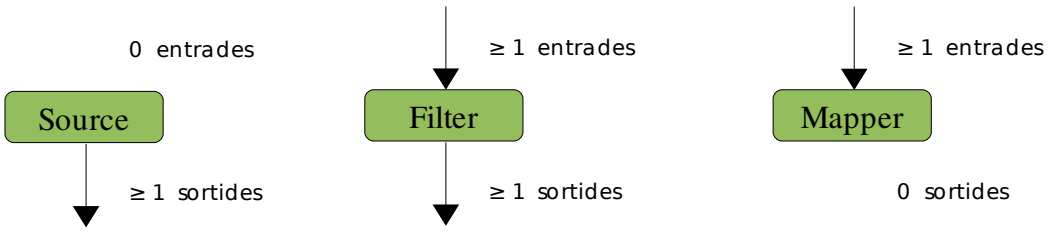


Figura 17: Source, Filter i Mapper

La sortida d'un filtre no és sempre del mateix tipus i per tant cal vigilar a l'hora de utilitzar-la com a entrada d'un altre filtre, ja que no sempre és possible.

5.4.1 Execució del pipeline

Com ja hem comentat anteriorment els *"process objects"* es connecten entre sí per formar el camí que seguiran les els *"data objects"* dades durant el seu tractament. Per obtenir un millor rendiment en el pipeline de visualització el que es fa és només tornar a calcular la sortida d'un filtre quan la seva entrada ha estat modificada.

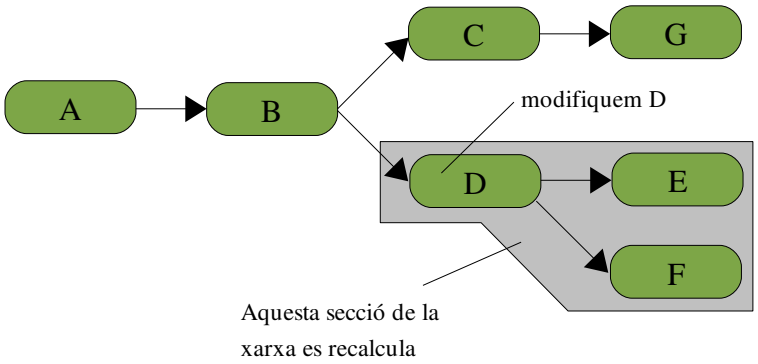


Figura 18: Pipeline de visualització

A la Figura 18 podem veure com la modificació de l'entrada de D provoca que s'hagi de tornar a calcular E i F ja que aquests filtres depenen de D. Però en canvi la resta de la xarxa no es veu afectada perquè la seva entrada no canvia.

5.4.2 Gestió de la memòria

Treballant amb imatge mèdica ja sigui per la mida d'aquestes o bé per la complexitat dels algorismes no és d'estranyar la utilització d'alguns centenars de MB de memòria. La optimització d'aquesta és molt i molt important.

El fet de treballar utilitzant un *pipeline* de visualització on es representa a les dades com un component que va viatjant a través dels diferents objectes que les tracten permet gestionar la memòria de dues formes.

Model estàtic

En aquest model si tenim varis filtres connectats entre si, les sortides generades pels filtres intermedis sempre es mantenen a memòria, de manera que si cal tornar a calcular el procés perquè s'ha produït algun canvi en la configuració d'algun algorisme de tractament de dades, només cal continuar en el punt on s'ha produït el canvi ja que l'entrada que necessita aquell filtre encara es troba present a memòria i no l'hem de tornar a generar.

Model dinàmic

Consisteix en guardar els resultats de les dades intermèdies del pipeline visualització, només fins que són necessàries. Aquesta utilització provoca una major càrrega computacional ja que cada vegada que els resultats són demanats s'ha de calcular tot el sistema perquè no es tenen el resultats intermedis que en permetrien fer el càlcul a partir d'aquell punt.

A grans trets podem dir que el model dinàmic redueix memòria a canvi de consumir temps de càlcul. I al revés, el model estàtic consumeix més memòria però estalvia temps de càlcul.

Amb el mètode *SetReleaseDataFlag(bool)* podem escollir si utilitzar el model dinàmic o el model estàtic de memòria. Per defecte s'utilitza el model dinàmic.

A l'exemple de la Figura 19 es pot veure com el model estàtic executa cada procés només una vegada guardant els resultats intermedis. En canvi en el model dinàmic com que va alliberant memòria l'execució de D requereix que es tornin a executar A i B.

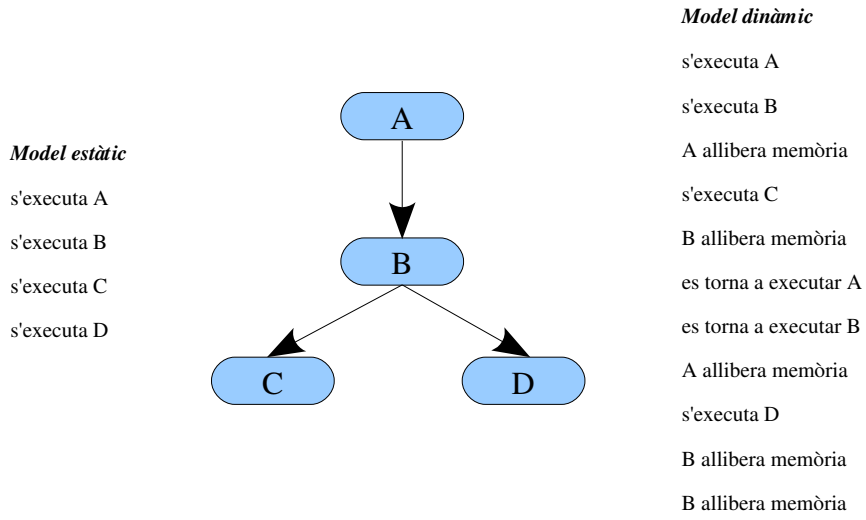


Figura 19: Execucions que es produeixen en el model estàtic i dinàmic

5.5 Coneixements previs sobre les llibreries

Els meus coneixements sobre les llibreries utilitzades eren nuls pel que fa a les VTK i a les ITK.

Les VTK em va costar saber-les utilitzar, però més que un problema de la llibreria, era que mai havia fet cap assignatura de gràfics. Em faltava molta terminologia bàsica i entendre el concepte d'escena, actors, llums, etc. Vaig haver de dedicar molt de temps entre llegir sobre visualització 3D i seguir exemples.

Les ITK tampoc les havia utilitzat mai i el cert és que la documentació que porten és una

mica pobre. L'alt grau d'utilització de *templates* que fan, va fer que al principi em costés molt seguir el codi del Starviewer.

I finalment les QT tot i haver-les utilitzat en el projecte de la tècnica i en una assignatura de la superior, al tractar-se de la versió més actual també em van obligar a dedicar un temps preciós per aclarir perquè coses que funcionaven a la versió anterior que jo sí havia utilitzat, a la nova versió no funcionaven.

5.6 La plataforma Starviewer

Com que aquest projecte forma part d'un projecte més gran anomenat Starviewer, primer va caldre entendre la part que ja estava feta i a la qual ens havíem de cenyir a l'hora d'integrar els nostres mòduls o classes. A la Figura 20 es mostra a grans trets el disseny de classes de l'aplicació proporcionada inicialment del Starviewer que va ser utilitzat com a punt de partida d'aquest projecte. A continuació es comenta el disseny inicial.

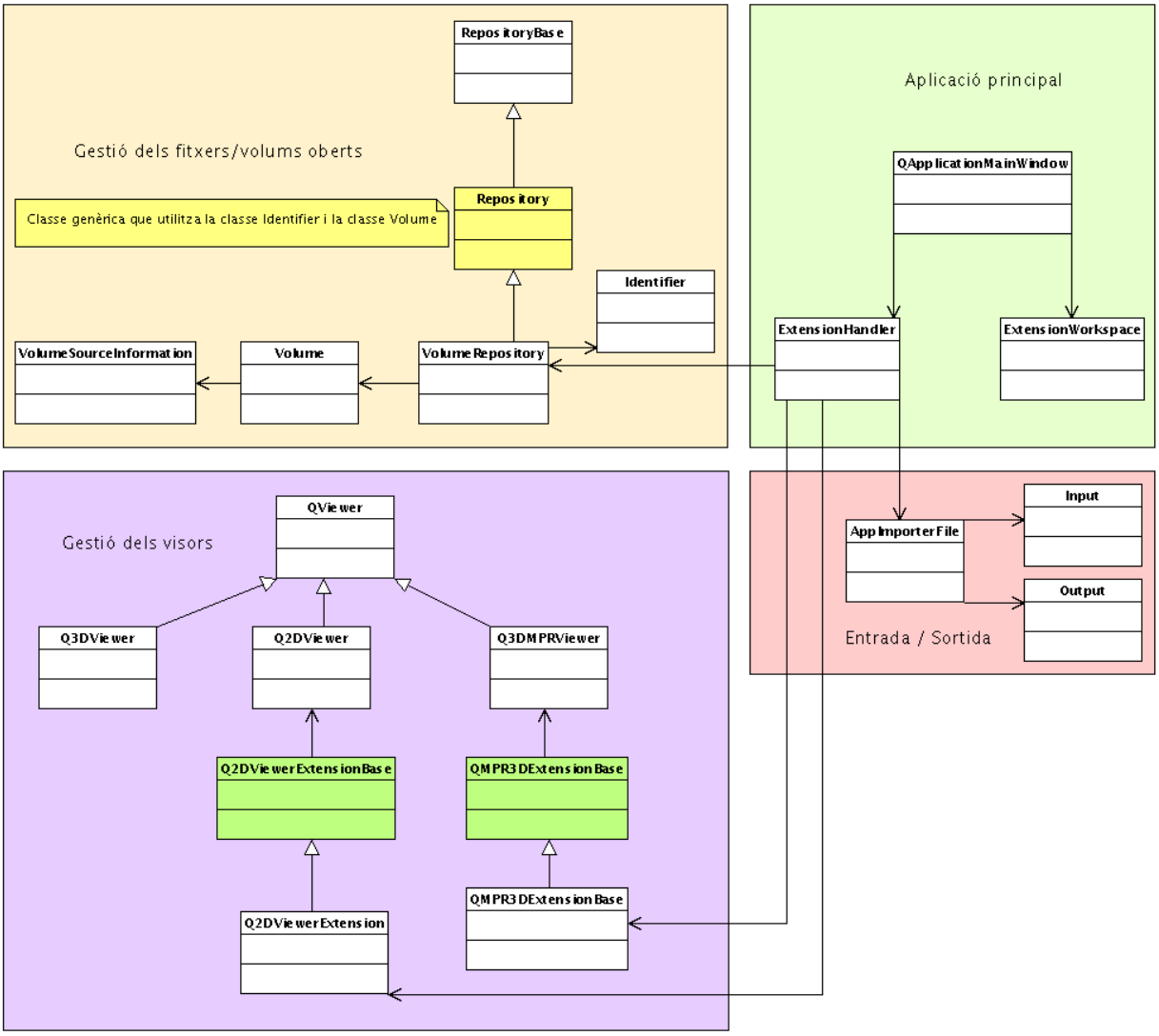


Figura 20: Disseny inicial del Starviewer simplificat

La notació utilitzada és la pròpia del UML i per tant les fletxes amb la punta blanca es corresponen a relacions de herència, les de la punta negra a la relació de utilitza. Les classes amb el fons verd tenen un correspondència directa amb interfícies de l'usuari. A partir d'ara tots els diagrames de classes que s'utilitzin mantindran aquesta notació.

S'han suprimit algunes classes per simplificar-ne el disseny, de fet l'aplicació estava formada per 45 classes. En concret s'han tret les classes que formen el patró factory i del patró singleton que intervenen a la creació de nous objectes i a l'hora d'executar una única instància del repositori de volums. També s'ha suprimit del diagrama les classes que aportaven funcionalitats a l'aplicació però que de cares a aquest projecte no afecten. Un gran nombre de classes hereten de QObject, per estalviar relacions que només fan que carregar el diagrama i no aportar informació per a la comprensió d'aquest, s'ha decidit suprimir-les també.

Les classes d'un mateix mòdul s'agrupen sota un requadre del mateix color. D'aquesta manera es pot observar que bàsicament tenim 4 mòduls:

5.6.1 Entrada / Sortida

Llegeix els fitxers de disc i els carrega a memòria o bé fa el procés invers permetent guardar a disc un model que tenim carregat a memòria. Aquesta segona funcionalitat està pensada per permetre la conversió entre diferents formats de fitxer.

5.6.2 Gestió dels fitxers i volums oberts

Quan s'obre un fitxer, independentment de si es tracta d'un seguit de llesques que col·locades degudament permeten generar un volum com si es tracta d'una única llesca, a memòria es guarda com un volum dins la classe *Volume*. Si la imatge conté informació associada, aquesta es guarda dins de *VolumeSourceInformation*. I tots els fitxers (*Volume*'s) oberts es gestionen a la classe *VolumeRepository* que és qui porta el control del fitxers que tenim oberts.

La classe pare de *VolumeReppository* és una classe genèrica anomenada *Repository* que s'instancia a partir d'un identificador de volum *Identifier* i un volum *Volume*.

5.6.3 Gestió dels visors

Per visualitzar una imatge es disposa de diferents visors, entenent per visor una finestreta que conté una zona on es mostra d'alguna forma el contingut del fitxer carregat i disposa d'uns botons pensats per a les operacions o consultes que es puguin fer des d'aquell visor. Un mateix model pot ser presentat de forma diferent depenent del visor utilitzat. A nivell de disseny tots els visors hereten de la classe *QViewer*. Les dues classes existents que en deriven actualment són Q2DViewer i Q3DMPRViewer. Aquestes dues classes encapsulen

les classes VTK que permeten mostrar les imatges llesca a llesca i la classe que defineix tres plans sobre el model de voxels de manera que es mostren els punts del model on intersequen els plans.

La classe *Q2DViewer* i *Q3DMPRViewer* s'integren dins una interfície d'usuari anomenades *Q2DViewerExtensionBase* i *Q3DMPRViewerExtensionBase* respectivament.

I finalment trobem les classes que hereten de les classes *Q2DViewerExtensionBase* i *Q3DMPRViewerExtensionBase*, que com que també hereten Qwidget poden ser incrustades dins l'aplicació principal.

5.6.4 Aplicació principal

La classe que defineix la pantalla principal i el seu comportament és *QapplicationMainWindow*. Aquesta classe conté una *ExtensionWorkspace* que és a on es van obrint els diferents visors d'aplicacions. Un visor s'obre des de una crida de la classe *QExtensionHandler*.

A la Figura 21 es vol mostrar gràficament a què es corresponen les classes que tenen una interpretació gràfica.

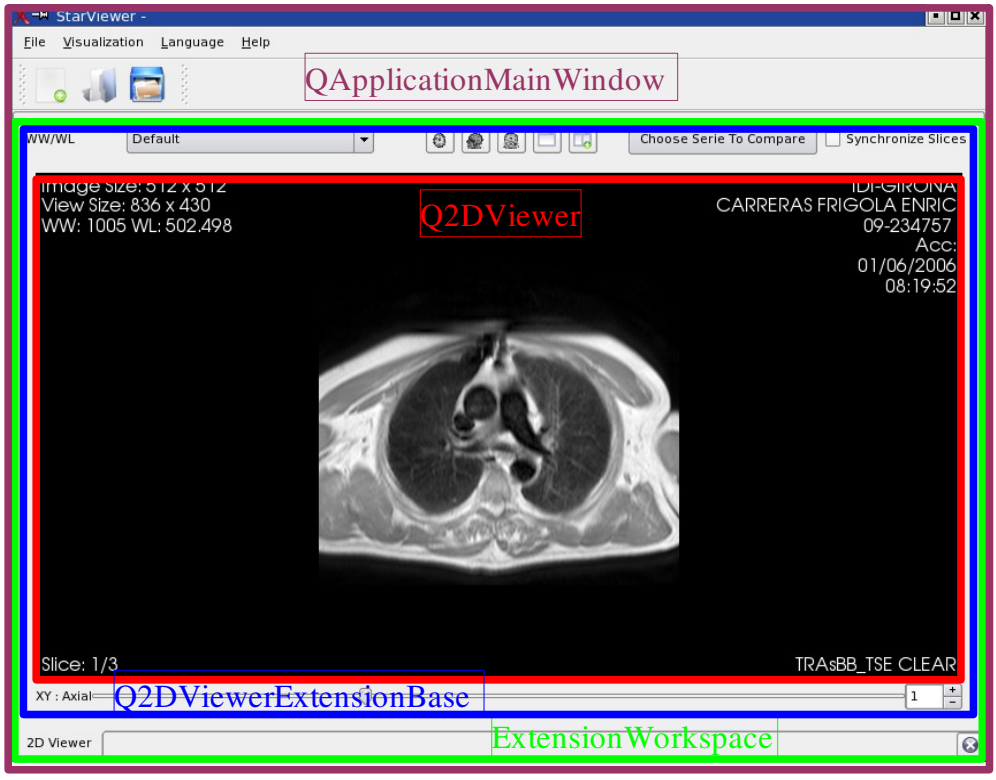


Figura 21: Correspondència gràfica d'algunes classes que apareixen al disseny del Starviewer

6 Anàlisi i disseny de l'aplicació

En aquest capítol presentem l'anàlisi de requeriments i el disseny de l'aplicació. Primer farem la descripció dels requeriments funcionals de l'aplicació i llavors el diagrama de casos d'ús i la fitxa de cada cas.

6.1 Anàlisi de requeriments

Com ja hem dit anteriorment, l'aplicació desenvolupada s'ha d'integrar en una plataforma de visualització d'imatges mèdiques, que en el moment de l'anàlisi permetia obrir un o més models de voxels. Per tant en l'anàlisi hem suposat que podem accedir a un conjunt de models de voxels carregats a memòria a través els mètodes que proporciona la plataforma.

6.2 Requeriments funcionals

Els requeriments funcionals que s'han tingut en compte en el desenvolupament d'aquest projecte són:

- L'aplicació ha de permetre la visualització de les imatges 2D. Interessarà que el metge pugui variar el contrast de la imatge per ressaltar el que més li convingui.
- Les imatges 2D que formin un dinàmic del cor s'hauran de poder reproduir i visualitzar com si d'un vídeo es tractés. Si es vol també s'ha de poder escollir reproduir només unes quantes imatges de manera repetitiva i començar la reproducció des d'on es vulgui.
- A partir d'una seqüència de imatges 2D que continguin el batec del cor ha de ser possible crear un fitxer de vídeo on es vegi de forma tant exacte com sigui possible la reproducció que s'aconsegueixi des del visor 2D.
- S'han de poder visualitzar models 3D, permetent que el metge esculli quines regions del model li interessa visualitzar.
- L'aplicació ha de permetre la visualització de models 3D del cor de forma gràfica, així com veure com evoluciona una llesca de la imatge al llarg del temps.
- S'haurà de poder prendre algun tipus de mesura com pugui ser la distància.

6.3 Diagrames de casos d'ús

Amb els diagrames de casos d'ús es mostrarà de manera visual com interactua l'usuari amb l'aplicació i què hi pot fer. S'han estructurat els diagrames de casos d'ús agrupant-los pels objectius marcats en aquest projecte. Cada subapartat segueix el mateix esquema, primerament la representació gràfica i esquemàtica dels casos d'ús i a continuació les fitxes corresponents a aquell diagrama.

6.3.1 Visualització 2D

En aquest apartat presentem el diagrama de casos d'ús de la visualització 2D (Figura 22) juntament amb la fitxa corresponent per cada cas d'ús.

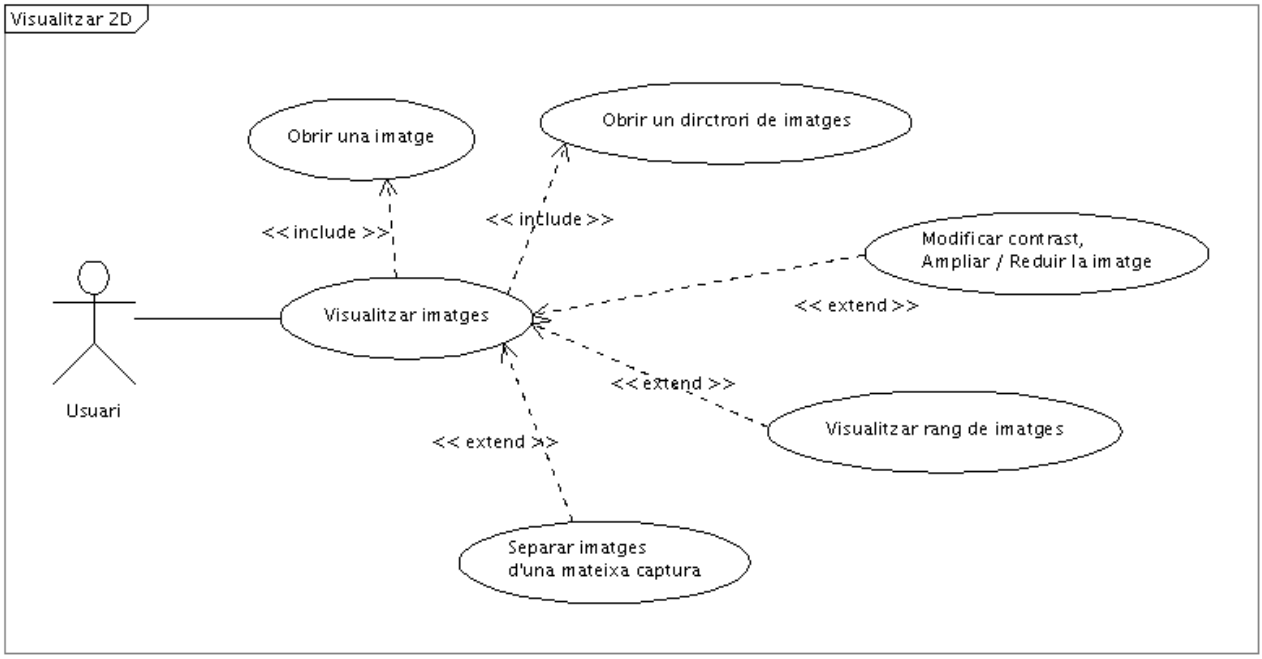


Figura 22: Casos d'ús de la visualització 2D

A continuació presentem la fitxa pel cas d'ús visualitzar imatges que és l'únic que tenim marcat dins l'objectiu de visualització 2D.

Cas d'ús: Visualitzar imatges	
Descripció	Permetrà visualitzar per pantalla les imatges 2D, tant les que provinguin d'un fitxer com les que es trobin en un directori
Actors	usuari
Precondició	Que el fitxer a obrir sigui DICOM o .mhd. En cas que sigui un directori cal que el seu contingut siguin DICOM's.
Flux Principal	1. Clicar damunt la icona obrir fitxer o bé obrir directori
Flux Alternatiu	1. Accedir-hi a través del menú *File / Open* o bé *File / Open DICOM directory*
Post Condició	Es mostra per pantalla el contingut del fitxer
Comentaris	En cas que s'obri un directori o un fitxer que contingui més d'una imatge, la interfície permet canviar de imatge

6.3.2 Visualització de 2D afegint-hi temps

En aquest apartat presentem el diagrama de casos d'ús de la visualització 2D afegint-hi temps (Figura 23), juntament amb la fitxa corresponent per cada cas d'ús.

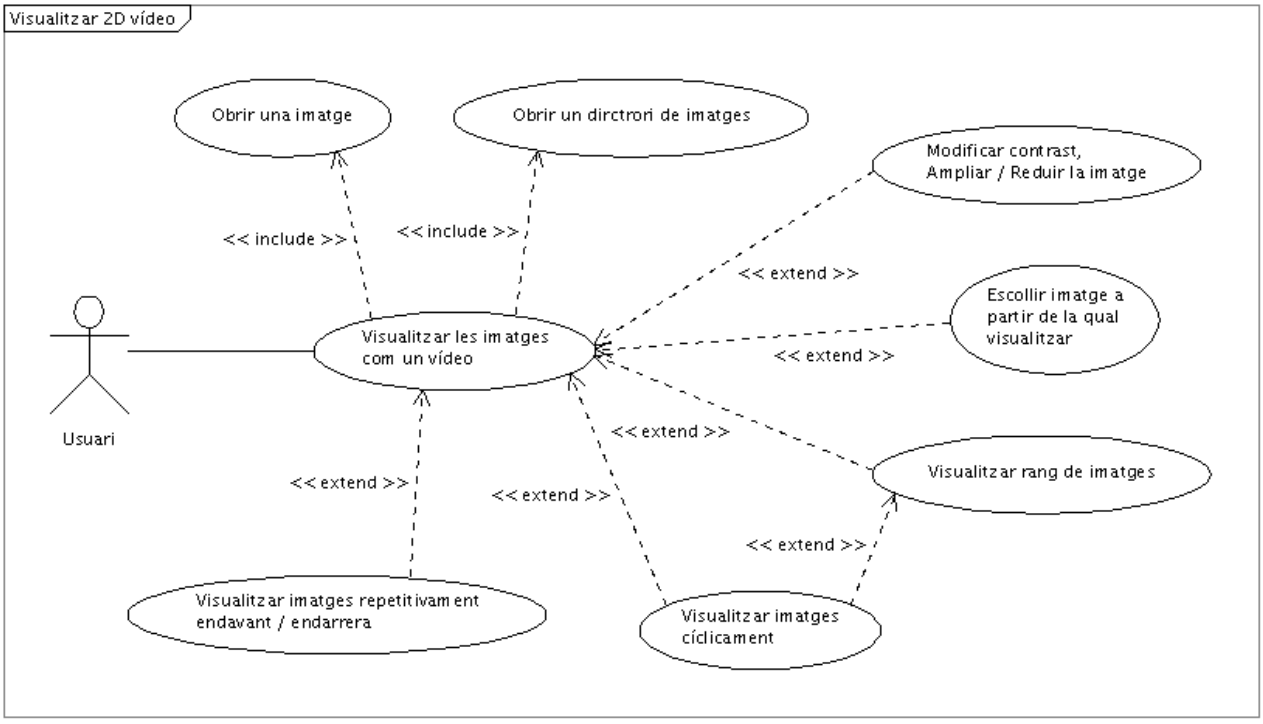


Figura 23: Casos d'ús de la visualització 2D afegint-hi temps

A continuació presentem la fitxa pel cas d'ús visualitzar imatges com un vídeo que és l'únic que tenim marcat dins l'objectiu de visualització 2D afegint-hi temps.

Cas d'ús: Visualitzar les imatges com un vídeo	
Descripció	Permetrà visualitzar un fitxer o directori de imatges com si es tractés d'una pel·lícula
Actors	usuari
Precondició	Que s'hagi obert un fitxer amb més d'una imatge o un directori d'imatges
Flux Principal	1. Escollir rang d'imatges que es vol visualitzar 2. Escollir el tipus de visualització cíclica 3. Clicar al botó play 4. Ajustar velocitat de reproducció
Flux Alternatiu	-
Post Condició	Es visualitza la reproducció
Comentaris	Durant la reproducció es poden ajustar les dimensions i el contrast de les imatges així com ampliar o reduir el nombre de imatges a reproduir.

6.3.3 Visualització 3D per plans

En aquest apartat presentem el diagrama de casos d'ús de la visualització 3D per plans (Figura 24), juntament amb la fitxa corresponent per cada cas d'ús.

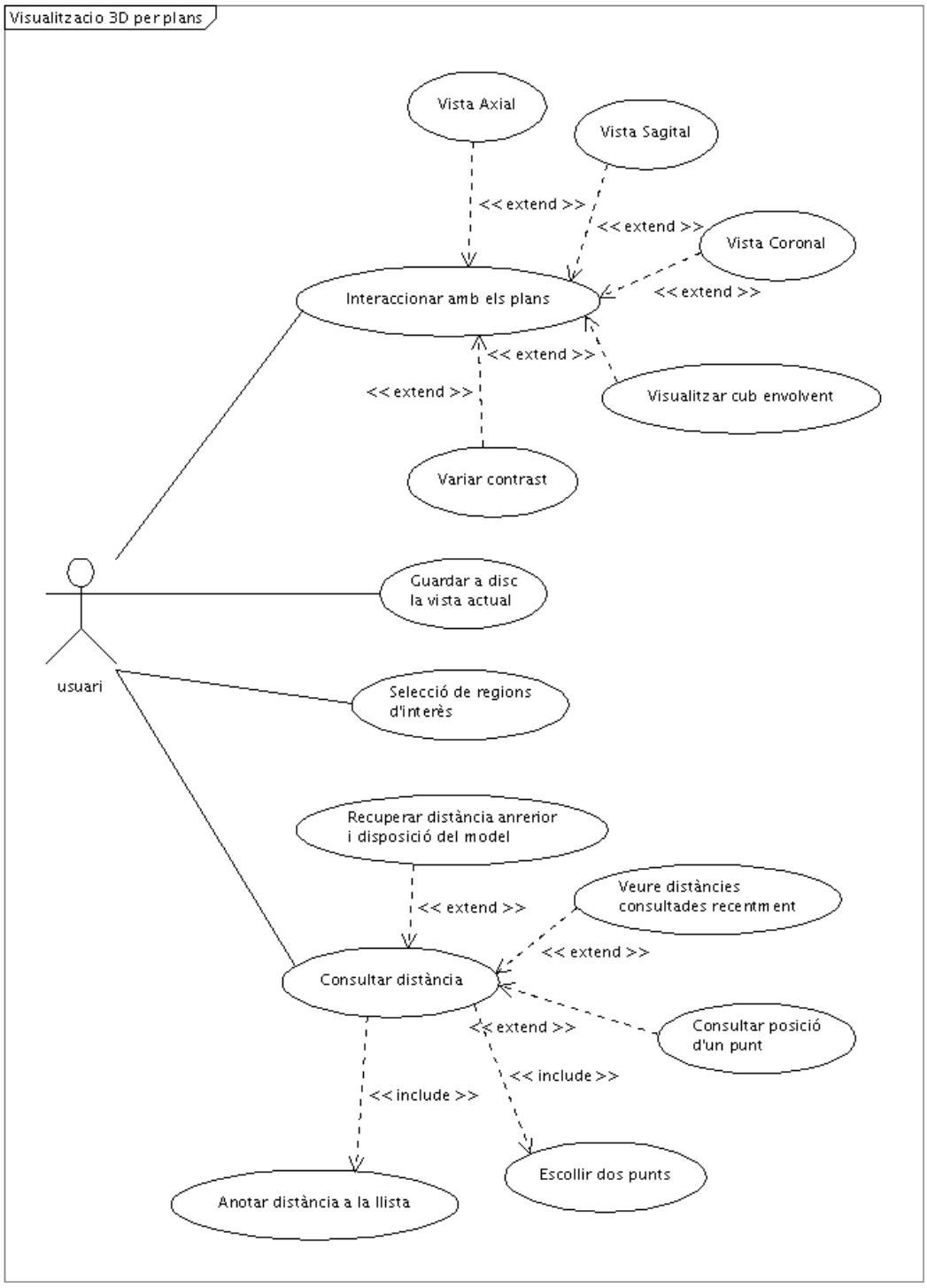


Figura 24: Casos d'ús de la visualització 3D per plans

A continuació presentem la fitxa que es correspon al 4 casos d'ús de la visualització 3D per plans.

Cas d'ús: Interaccionar amb els plans	
Descripció	Es mostren 3 plans ortogonals a la pantalla que permeten visualitzar la superfície sobre la qual estiguin col·locats, de manera que si els col·loquem al mig del volum veiem el que veuríem si partíssim el volum per aquell pla
Actors	usuari
Precondició	Que s'hagi obert un model 3D amb el visor 3D per plans
Flux Principal	1. Clicar damunt del pla desitjat amb el botó central del ratolí 2. Manté el botó premut i mou el ratolí
Flux Alternatiu	1. Clicar amb el botó dret damunt el pla 2. Desplaçant-se en horitzonta varia el window i en vertical el level
Post Condició	L'usuari pot veure el "mateix" que veuria si tingués el cor a davant i el pogués seccionar per la zona on està situat el pla
Comentaris	La interacció permet traslladar i rotar els plans en vàries direccions, a més de variar el contrast de les imatges permet mostrar un rectangle que en volta el volum, i col·locar el model amb unes posicions preprogramades. Aquestes posicions són vista axial, sagital i coronal.

Cas d'ús: Guardar a disc la vista actual	
Descripció	Guardar en format .jpg el que l'usuari està veient del model
Actors	usuari
Precondició	Que s'hagi obert un model 3D amb el visor 3D per plans
Flux Principal	1. Clicar sobre la icona de salvar 2. Donar un nom al fitxer 3. Escollir on es guardarà
Flux Alternatiu	-
Post Condició	S'obté un fitxer amb extensió jpg que conté una imatge del que l'usuari estava veient.
Comentaris	-

Cas d'ús: Consultar distància	
Descripció	Permet mesurar distàncies en el plans ubicats al model
Actors	usuari
Precondició	Que s'hagi obert un model 3D amb el visor 3D per plans
Flux Principal	1. Clicar sobre la icona de mesurar distàncies 2. Clicar sobre el model a on situem el punt origen 3. Repetir pas anterior per ubicar el punt destí
Flux Alternatiu	1. Clicar damunt la llista de distàncies mesurades anteriorment
Post Condició	La última distància calculada es visualitza a la cantonada de la imatge. En cas de recuperar una distància prèviament calculada el plans es col·loquen amb la mateixa disposició de quan s'ha pres la mesura.
Comentaris	Cada distància que es calcula queda anotada a una taula lateral. Per tal de saber de forma ràpida en quin pla s'ha calculat es pinta l'entrada a la taula del mateix color que el contorn del pla.

6.3.4 Visualització 4D per plans

En aquest apartat presentem el diagrama de casos d'ús de la visualització 4D per plans (Figura 25), juntament amb la fitxa corresponent per cada cas d'ús.

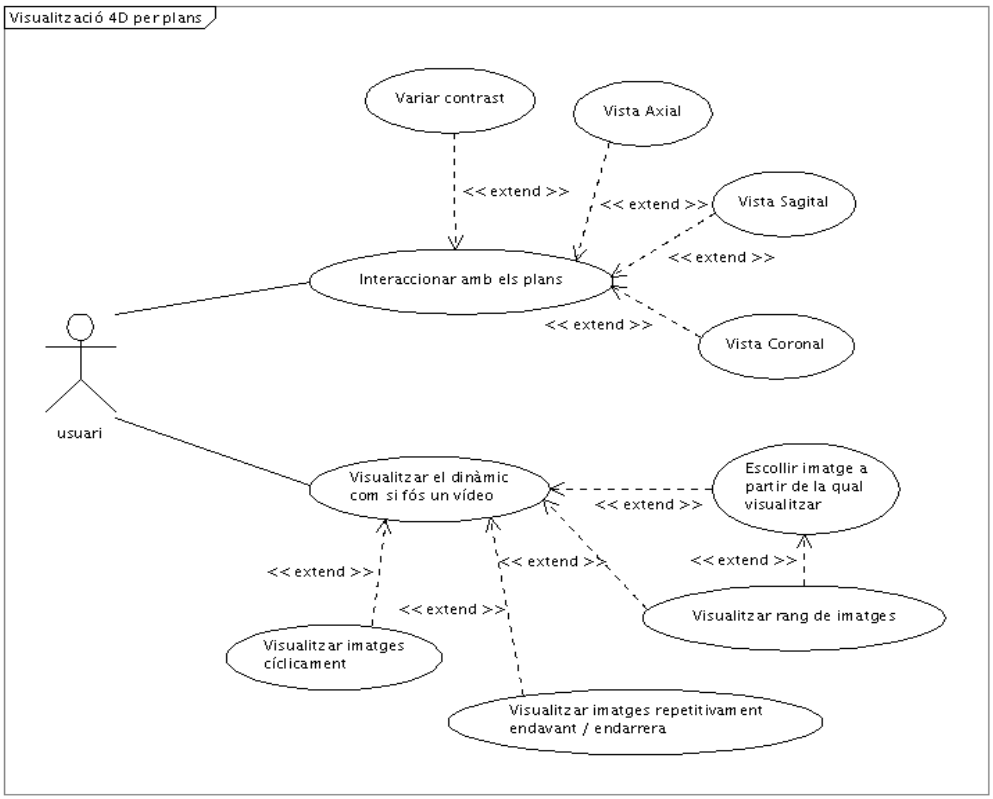


Figura 25: Casos d'ús de la visualització 4D per plans

A continuació es presenta la fitxa que es correspon a cada cas d'ús referent a la visualització 4D per plans.

Cas d'ús: Interaccionar amb els plans	
Descripció	Es mostren 3 plans ortogonals a la pantalla que permeten visualitzar la superfície sobre la qual estiguin col·locats
Actors	usuari
Precondició	Que s'hagi obert un model 3D amb el visor 3D per plans
Flux Principal	1. Clicar damunt del pla desitjat amb el botó central del ratolí 2. Manté el botó premut i mou el ratolí
Flux Alternatiu	1. Clicar amb el botó dret damunt el pla 2. Desplaçant-se en horitzontal varia el window i en vertical el level
Post Condició	L'usuari pot veure el "mateix" que veuria si tingués el cor a davant i el pogués seccionar per la zona on està situat el pla
Comentaris	La interacció permet traslladar i rotar els plans en vàries direccions, a més de variar el contrast de les imatges permet mostrar un rectangle que envolta el volum, i col·locar el model amb unes posicions preprogramades. Aquestes posicions són vista axial, sagital i coronal.

Cas d'ús: Visualitzar el dinàmic com si fos un vídeo	
Descripció	A través del botons de la interfície es pot reproduir el model
Actors	usuari
Precondició	Que s'hagi obert un model 3D amb el visor 3D per plans
Flux Principal	1. Escollir rang d'instants de temps que es vol visualitzar el model 2. Escollir el tipus de visualització cíclica 3. Clicar al botó play 4. Ajustar velocitat de reproducció
Flux Alternatiu	-
Post Condició	Al tractar-se de la visualització d'un mateix models en diferents instants de temps dóna la sensació de moviment
Comentaris	-

6.3.5 Visualització 3D

En aquest apartat presentem el diagrama de casos d'ús de la visualització 3D (Figura 26), juntament amb la fitxa corresponent per cada cas d'ús.

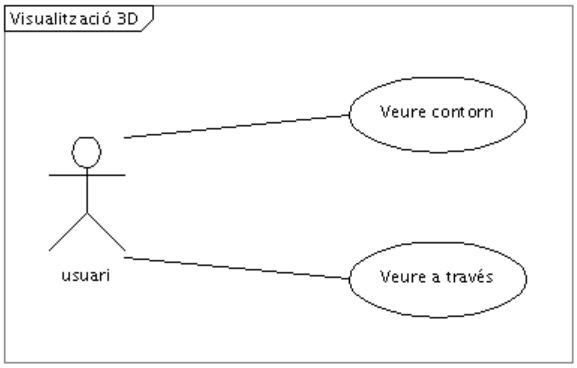


Figura 26: Casos d'ús de la visualització 3D

Tot seguit podem veure les fitxes que es corresponen a cada cas d'ús de la visualització 3D.

Cas d'ús: Veure contorn	
Descripció	Mostra tridimensionalment les parets del cor
Actors	usuari
Precondició	Que s'hagi obert un model 3D amb el visor 3D
Flux Principal	1. Escollir IsoSurface
Flux Alternatiu	-
Post Condició	Apareix en 3 dimensions els teixits del cor preprogramats
Comentaris	Permet fer zoom el model i girar-lo utilitzant el ratolí

Cas d'ús: Veure volum	
Descripció	Mostra tridimensionalment el cor
Actors	usuari
Precondició	Que s'hagi obert un model 3D amb el visor 3D
Flux Principal	1. Escollir Ray Casting
Flux Alternatiu	1. Escollir MIP3D
Post Condició	Apareix en 3 dimensions el volum del cor
Comentaris	Permet fer zoom el model i girar-lo utilitzant el ratolí

6.3.6 Exportar a fitxer de vídeo

En aquest apartat presentem el diagrama de casos d'ús de la visualització 3D (Figura 27), juntament amb la fitxa corresponent per cada cas d'ús.

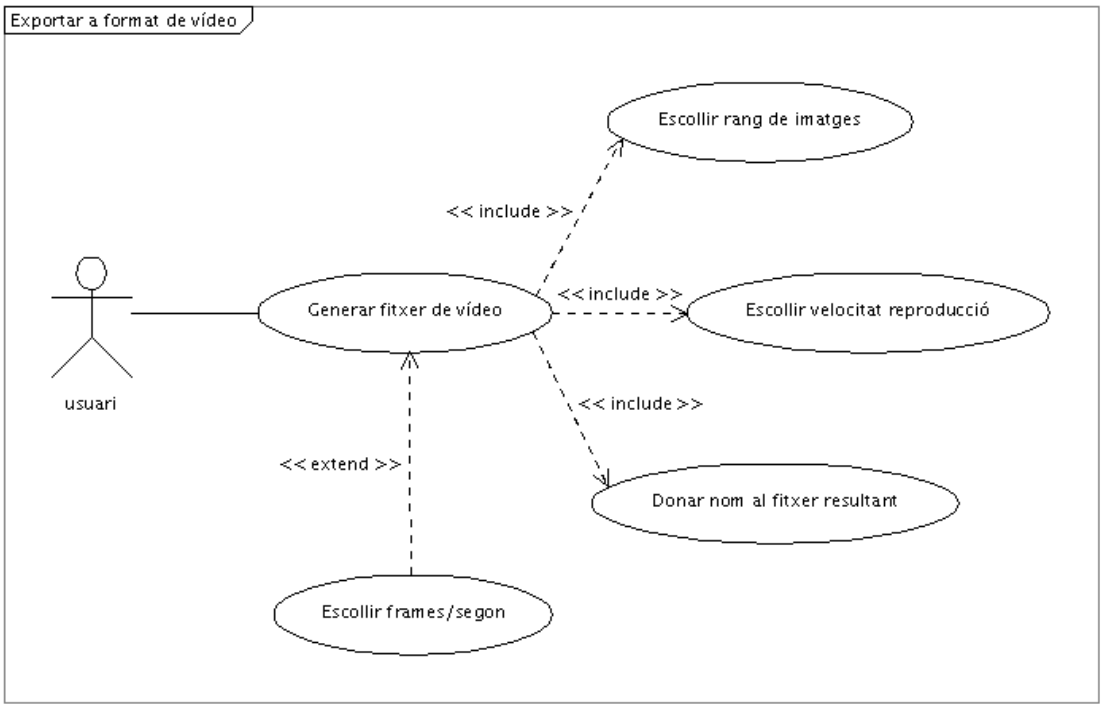


Figura 27: Casos d'ús d'exportar a format de vídeo

Tot seguit podem veure la fitxa que es correspon a l'exportació a fitxer de vídeo.

Cas d'ús: Exportar a fitxer de vídeo	
Descripció	Generar un fitxer resultat que conté una seqüència de imatges reproduibles
Actors	usuari
Precondició	Que s'hagi obert un model amb el visor 2D
Flux Principal	1. Escollir rang de imatges 2. Triar velocitat de reproducció 3. Clicar damunt el botó de gravació 4. Escollir tipus de vídeo generat 5. Donar ubicació i nom al fitxer
Flux Alternatiu	-
Post Condició	S'obté un fitxer de vídeo
Comentaris	Els diferents tipus de vídeo que es poden generar fan referència a les imatges per segon i a si el vídeo és entrellaçat o progressiu.

6.4 Diagrama de classes

A la Figura 28 podem veure com ha quedat el Starviewer un cop acabat aquest projecte i afegides les millores proposades com a objectius d'aquest projecte. Perquè no hi hagi confusió sobre la feina feta s'ha simbolitzat **de color blau les classes que han patit ampliacions o modificacions, i de color vermell les classes noves que s'han creat**. Vegis també que s'ha afegit un nou mòdul anomenat **Exportar vídeo** que engloba un conjunt de classes i per simplicitat s'ha decidit obviar-les i explicar-les amb detall a l'apartat 8.

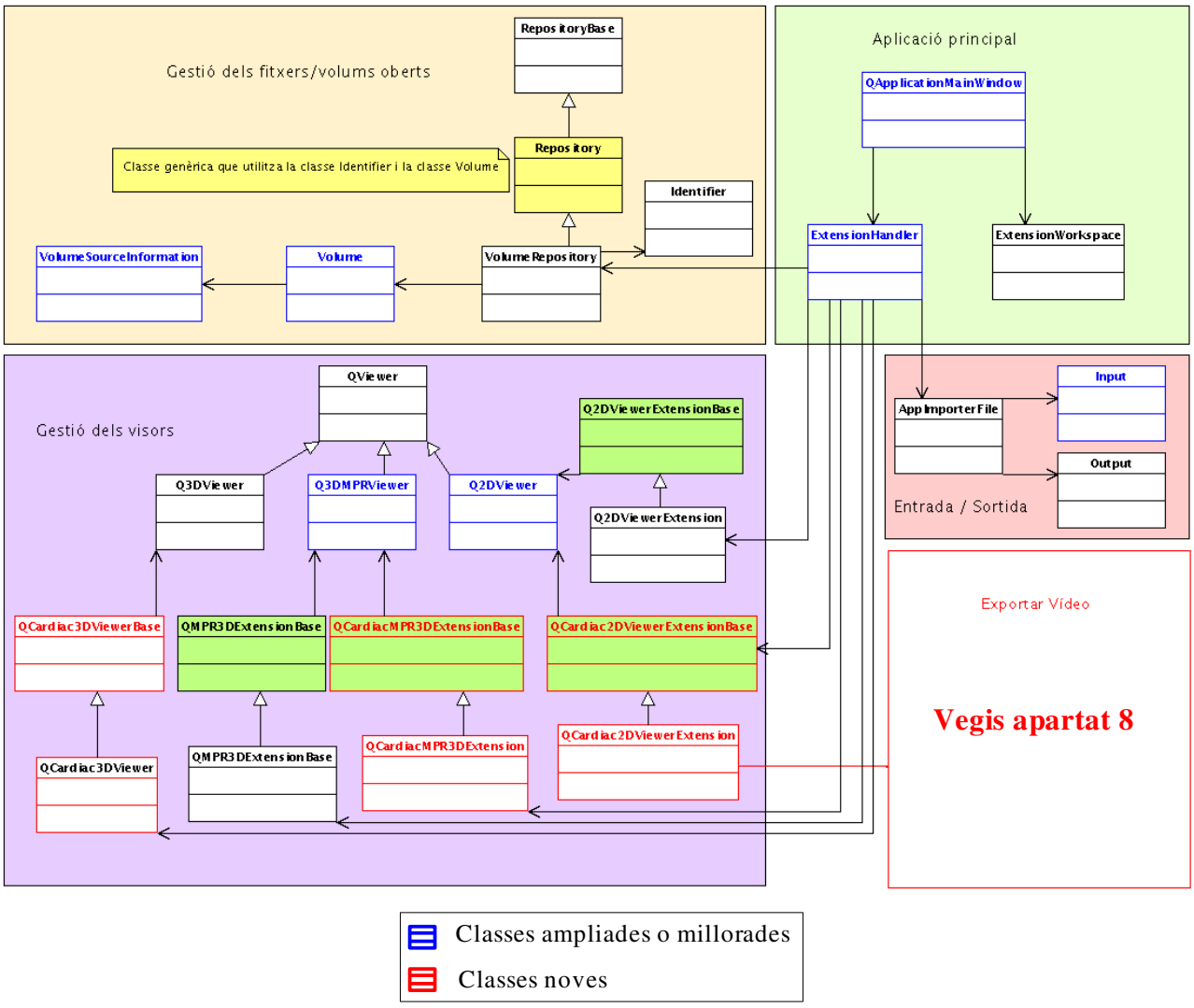


Figura 28: Diagrama de classes del Starviewer en acabar aquest projecte

Les principals classes afegides a la plataforma i ressaltades amb vermell a la Figura 28 són les que ens permeten acomplir els objectius marcats a l'inici d'aquest projecte. Aquestes classes són:

- QCardiac2DViewerExtensionBase i QCardiac2DViewerExtension per la visualització

de imatges de dues dimensions.

- QCardiacMPR3DExtensionBase i QCardiacMPR3DExtension per a la visualització 3D utilitzant plans.
- QCardiac3DViewerBase i QCardiac3DViewer per la visualització 3D.
- Exportar Vídeo que es tracta d'un mòdul que agrupa un conjunt de classes relacionades.

Destacar també que algunes millores han implicat fer canvis a classes ja existents que com hem dit anteriorment es veuen de color blau a la Figura 28.

L'explicació més detalla de cada classe i de les modificacions realitzades es veurà en els següents capítols.

7 Visor 2D de dades de cor

Pel que fa a la visualització de imatges 2D, en aquest projecte s'havien marcat dos objectius clars:

- Visualitzar imatges 2D
- Reproduir imatges 2D

Ambdós objectius s'han construït sota una mateixa finestra que és el visor 2D. El visor 2D permet mostrar les diferents llesques que pugui contenir un fitxer, o bé obrir un directori que contingui varis fitxers DICOM.

7.1 Disseny

El disseny d'aquest visor ha implicat crear noves classes i ampliar funcionalitats d'algunes ja existents. Les classes noves s'han afegit en el mòdul que hem anomenat *Gestió de visors* a la Figura 20 de l'apartat 5.6, i els mòduls que han vist ampliades i millorades les seves classes són el mòdul *Entrada/Sortida* i el de *Gestió dels fitxers/volums oberts* de la mateixa figura.

7.1.1 Millores en el disseny

Les millores han consistit en crear el visor pròpiament dit amb totes les seves funcionalitats. A la Figura 31 podem veure'n el seu disseny de classes.

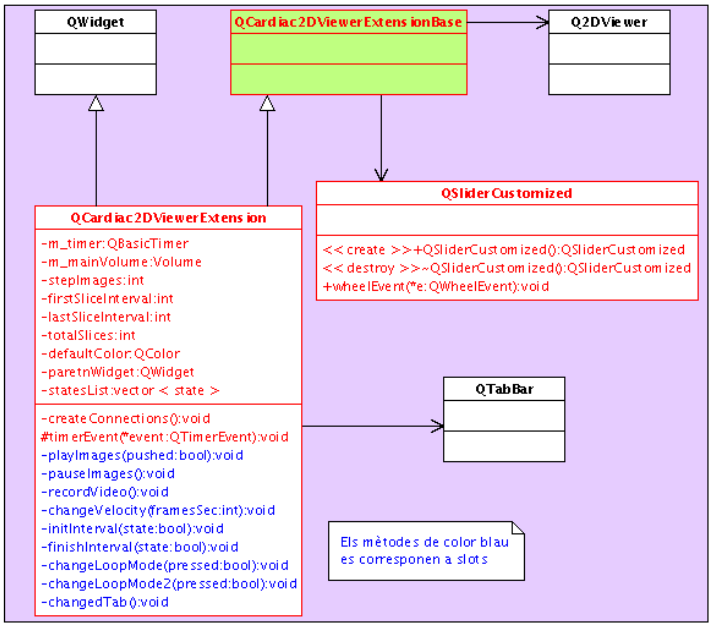


Figura 29: Disseny de classes del visor 2D

L'explicació que es correspon al funcionament de cada classe es troba a continuació:

- **QcardiacViewer2DExtensionBase**

  És la interfície d'usuari dissenyada amb *QTDesigner*, incorpora entre d'altres, botons de control i la classe *Q2DViewer* que a través de les VTK permet mostrar les imatges. L'altre classe que conté és la *QSliderCustomized* que deriva de *QSlider* i més endavant es comenta amb més detall.

- **Q2DViewer**

  Mostra les imatges a partir de la classe *vtkImageViewer2* i incorpora tot una sèrie de característiques com són, fer zoom de la imatge o ajustar-ne els nivells de contrast al nostres gust si els que es carreguen per defecte no són útils. Aquestes característiques de la imatge es modifiquen utilitzant la rodeta del ratolí, fent-la girar per fer el zoom i mantenint-la premuda per moure'ns al llarg i ample de la imatge.

- **QSliderCustomized**

  Un *slider* és l'element de la interfície que podem veure a la Figura 30. El *slider* que ens proporciona les QT (*QSlider*) té un comportament envers la rodeta del ratolí que no és el desitjat.

Figura 30: Slider

  Quan es mou un slider fent rodar la rodeta aquest es desplaça cap un costat o cap a l'altre en funció del sentit en què girem la rodeta. Però la barreta avança o retrocedeix fent salts de més d'un valor, (normalment tres posicions que és la configuració per defecte de la rodeta del ratolí). Com que ens interessava poder incrementar o decrementar d'un en un es va decidir crear una classe filla de la *QSlider* i programar l'esdeveniment *QwheelEvent*. D'aquesta manera es pot utilitzar el botó central del ratolí per visualitzar de forma còmoda cada llesca que composa la imatge, i es poden fer moviments més precisos que no pas arrastrant la barreta.

- **QCardiacViewer2DExtension**

  És la classe encarregada de la reproducció de les imatges així com de permetre la visualització de les diferents llesques presents en un mateix fitxer o directori que tinguem obert. Si les imatges que visualitzem provenen de fitxers DICOM i no totes les imatges tenen la mateixa orientació, aquesta classe ens proporciona pestanyes que agrupen les imatges que es corresponen a una mateixa orientació.

La classe que ens proporciona les pestanyes és la *QTabBar*. Com es pot veure és un element de la interfície que no s'ha afegit directament a la classe *QCardiac2DViewerExtensionBase* perquè el *QT Designer* no ens ho permetia.

La classe *QcardiacViewer2DExtension* destaca per tenir un gran nombre de mètodes slots que s'encarreguen de:

- Reproduir imatges
- Parar la reproducció
- Limitar el nombre de imatges que es vol reproduir
- Reproduir cíclicament les imatges, ja sigui endavant i enrere o tornant a començar quan s'ha arribat al final.

Per a la reproducció de imatges el que s'ha fet és capturar el senyal que emet un *timer* periòdicament i actuar en funció dels botons de la interfície que l'usuari ha premut o estan premuts. Mentre s'està reproduint un vídeo l'usuari pot avançar retrocedir activar o desactivar les opcions de repetició i limitar el nombre de imatges a reproduir.

Cal destacar el fet que tot i que les QT posseeixen *timers* de més bon utilitzar, ens vam decidir per aquest per ser la classe més lleugera de totes.

7.1.2 Ampliacions en el disseny

Ha estat necessari ampliar la classe *Input* (Figura 31) de manera que llegís les etiquetes de la capçalera dels DICOM que ens diferenciaven quan una mateixa seqüència de imatges present en un directori presentava imatges captades des d'un altre punt de vista.

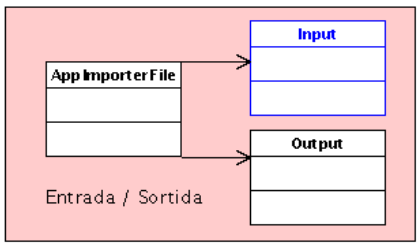


Figura 31: Mòdul de gestió entrada/sortida de l'aplicació

Al mateix temps aquesta etiqueta llegida del DICOM s'ha hagut de guardar a la classe *VolumeSourceInformation* (Figura 32) per si més endavant era requerida, de manera que també s'ha hagut d'ampliar les funcionalitats d'aquesta classe.

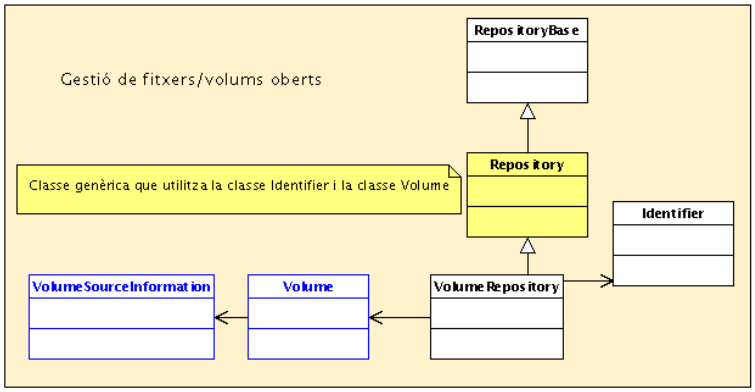


Figura 32: Mòdul que gestiona els fitxers/Volums oberts

Evidentment per afegir aquest nou visor a l'aplicació també ha implicat fer modificacions a la classe *QapplicationMainWindow* per afegir una nova entrada al menú i a la classe *ExtensionHandler* perquè quan es fes clic en el menú es mostrés aquest visor.

La classe Q2DViewer també va patir una lleugera modificació degut a què la versió de les VTK amb la qual es va desenvolupar aquest projecte era la més recent de totes i hi havia un *signal* de les QT que requeria més paràmetres que en la versió anterior.

7.2 Funcionalitats del visor 2D

Tal i com es pot veure a la Figura 33 es caracteritza per una zona central on es mostren les imatges i una sèrie de botons i pestanyes. Amb aquests botons podrem reproduir les llesques com si d'una pel·lícula de vídeo es tractés. Això és així ja que hi ha determinades proves on les llesques que es capturen estan pensades per ser visualitzades en moviment.

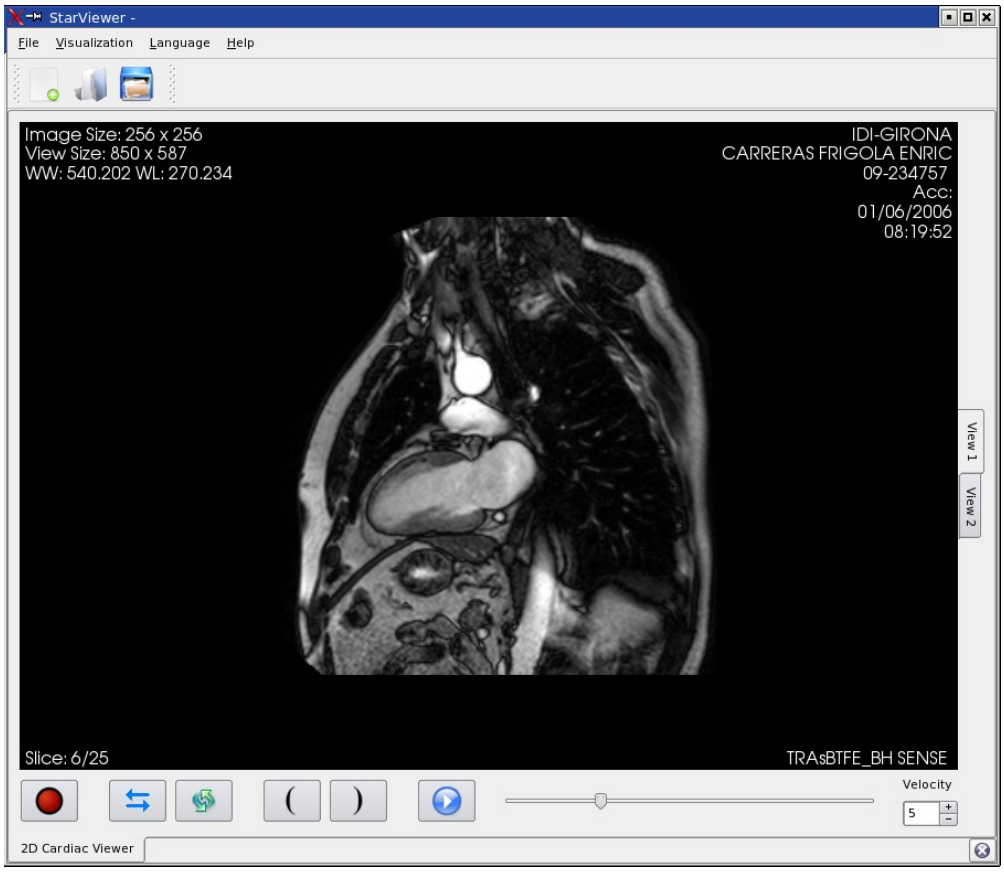


Figura 33: Visor 2D

Amb els controls es poden reproduir les llesques de principi a fi, de forma indefinida o bé repetitivament endavant i enrere, que són les funcionalitats que busquen els metges en aplicacions d'aquest tipus. A més a més permet regular la velocitat a la qual volem reproduir les llesques utilitzant el botó rodeta del ratolí damunt l'indicador de velocitat de reproducció. La velocitat de reproducció està expressada en imatges per segon.

En cas de desitjar visualitzar només un subconjunt de llesques també és possible fer-ho utilitzant els botons representats amb un parèntesis obert i un parèntesis tancat. La forma de visualitzar un subconjunt de imatges, és d'allò més senzilla. S'utilitza la barreta de desplaçament (*slider*) i ens situem a la primera llesca que volem visualitzar, llavors

premem el botó d'obrir parèntesis. El mateix si volem acotar el rang de imatges per la part superior, però prement el botó de tancar parèntesis. Per tal que l'usuari sigui conscient que té activada la opció de només visualitzar un subconjunt de llesques, el *slider* canvia de color i es torna blau.

El *slider* és d'allò més útil a l'hora d'escollir el rang de imatges que es vol visualitzar ja que permet ser arrastrat per saltar moltes llesques de cop i també permet utilitzar la rodeta del ratolí per passar les llesques d'una en una gràcies a les millores que li hem fet.

El visor 2D és qui incorpora la interfície per exportar les reproduccions de imatges a format de vídeo. En el següent capítol s'explica amb tot detall tot el que fa referència a l'exportació a format de vídeo i com s'ha integrat dins el visor 2D.

8 Exportar a altres formats

Un dels objectius d'aquest projecte era la possibilitat d'exportar els fitxers originals a d'altres formats. Com que la versió oficial del Starviewer ja suporta l'exportació de fitxers estàtics com és passar de DICOM a jpg, es va decidir treballar amb la generació de fitxers de vídeo. Per a tal propòsit s'ha optat per generar un fitxer resultant que conté una seqüència de vídeo comprimida utilitzant un compressor MPEG2.

Els avantatges de generar aquest fitxer són varis:

- Permet reproduir el fitxer sense necessitat d'utilitzar l'aplicació Starviewer. Amb una aplicació de reproducció de vídeo normal i corrent que que suporti MPEG2 en fem prou.
- El format DICOM contempla la possibilitat de guardar fitxers de vídeo MPEG2 al seu interior. Això fa que els vídeos puguin tenir les capçaleres del DICOM, amb la qual cosa podem associar un fitxer de vídeo a un pacient inequívocament i integrar-ho perfectament.
- Reproduir el fitxer, pot ser que tingui un cost computacional més baix que visualitzar-lo utilitzant el Starviewer. Per tant, pot ser que sigui una opció vàlida si es pretén reproduir més d'una seqüència de imatges al mateix temps i la reproducció de imatges que ofereix el Starviewer no suporta la càrrega computacional que li representa.

8.1 El compressor de MPEG2

Per crear un fitxer de vídeo a partir d'una seqüència de imatges cal un compressor que transformi les imatges en un flux MPEG2. La programació d'aquest compressor és una tasca feixuga i difícil on hi intervenen forces conceptes matemàtics i picaresca variada per tal d'optimitzar-ne la qualitat així com el temps de compressió. Com que programar-lo ens portaria molt de temps i no era l'objectiu d'aquest projecte es va decidir buscar a Internet a veure què hi havia fet que fos de codi lliure i que no ens lligués amb cap plataforma en concret, tot seguint la filosofia del Starviewer.

Els nombre de compressors trobat no va ser gens esperançador, sinó tot el contrari. En un principi es va contemplar la possibilitat d'utilitzar la llibreria *libavcodec* que forma part del

compressor *ffmpeg*.

Però la falta de documentació i la multitud d'opcions de què disposava en feien difícil tant comprensió com la utilització i per tant es va decidir utilitzar el compressor que es troba a la pàgina de la gent del Motion Picture Experts Group [MPEG2]. És un compressor [COMP212] fet a tall d'exemple de com ha de ser una possible implementació d'un compressor de MPEG2. Com que era simple de fer funcionar i el codi no era excessivament extens es va decidir integrar aquest compressor dins el Starviewer.

8.2 Funcionament del compressor

El que realitzava aquest compressor era a partir d'un fitxer d'entrada de paràmetres se li indicava quines propietats havia de tenir el fitxer resultant, i a partir de quins fitxers l'havia de generar. El nombre de formats de fitxers d'entrada era molt limitat. I el que es buscava era que a partir de qualsevol imatge que es pogués visualitzar amb l'Starviewer, es pogués dir al compressor quins eren els paràmetres desitjats per tal de generar el fitxer resultant.

El compressor presentava una vintena de paràmetres els quals es podien modificar per variar les característiques del fitxer de vídeo obtingut. Es va fer una tria de quines eren les funcionalitats més interessants de cares a la generació de vídeo a partir de imatges mèdiques i es va deixar la resta amb el seu valor per defecte.

Les opcions que es va decidir suportar i permetre que l'usuari escollís fan referència ales característiques del vídeo generat, a continuació les comentem:

- Vídeo entrellaçat o progressiu

  El vídeo entrellaçat a cada imatge que es mostra a la pantalla només es mostren les línies parells o imparells que composen la imatge. Si es mostren les línies parells les línies parells són les de la imatge anterior. I si es mostren les imparells les que són de la imatge anterior són les parells. Pot semblar un comportament estrany però així és com funciona la senyal de la televisió analògica. El vídeo progressiu és la solució més òbvia, cada imatge que es mostra és sencera i es correspon a la captura realitzada en un mateix instant de temps.
- Vídeo de baix cost computacional a l'hora de ser reproduït o normal

  En la generació de vídeo mpeg2, les imatges que acaben generant el flux de vídeo, poden ser de 3 tipus. El que s'ha fet ha estat permetre que el vídeo generat no tingués imatges del tipus B de manera que el vídeo ocupa més però no consumeix

tants recursos a l'hora de ser reproduït. Per una millor comprensió llegir l'annex B.

- Vídeo a 25, 30, 50 o 60 imatges / segon

  Per donar la sensació de moviment cal mostrar les imatges a una velocitat mínim de 25 imatges per segon. Hi ha dos sistemes àmpliament estesos en la generació de vídeo, el sistema PAL i el NTSC. Cada un dicta una mida de les imatges en píxels, unes proporcions, un nombre de imatges per segon, etc. El que s'ha pretès és seguir al màxim aquests dos sistemes per ser els més comuns en tot tipus d'aparells de reproducció.

  Es pot escollir entre generar vídeo entrellaçat a 25 imatges/segon, vídeo entrellaçat a 30 imatges/segon, vídeo progressiu a 50 imatges/segon i finalment vídeo progressiu a 60 imatges/segon. El fet de permetre aquestes velocitats i no unes altres és per compatibilitat amb els sistemes PAL i NTSC.

8.3 Adaptació del compressor

El compressor utilitzat presentava una sèrie de problemes amb els quals es va haver de tractar. Primer es va comprovar que efectivament funcionava i que els resultats eren visibles per diferents reproductors de vídeo. Un cop assegurat que el codi feia el que havia de fer es va comprovar si hi havia fugues de memòria, utilitzant l'eina Valgrind [VALGRIND], també de codi lliure.

El resultat va ser que efectivament hi havia memòria que es reservava i que no s'alliberava en finalitzar l'aplicació. Es va corregir aquest problema que si bé no tenia massa importància si el codi s'executava independentment, sí que en tenia, si el codi s'incloïa dins del Starviewer, on el compressor podria ser cridat més d'una vegada i finalment exhaurir la memòria. També es va fer que les matrius amb valors predefinits no fossin estàtiques i que només tinguessin vigència en cas de requerir la funcionalitat i mentre aquesta durés.

El següent pas va consistir en portar el codi al paradigma de l'orientació a objectes, ja que el codi es trobava en C, i l'aplicació que l'havia de utilitzar estava completament enfocada a la POO i utilitzava C++. Per solucionar alguns dels errors introduïts al fer la conversió es va utilitzar el el gdb [GDB] des de l'entorn gràfic que proporciona el KDevelop. En aquest pas va ser on van sorgir les dificultats més grans, ja que la conversió no era directa i de fet el resultat final es veu fortament condicionat al disseny inicial fet amb C. Així per exemple tenim una classe anomenada Global que és utilitzada per la majoria de classes i no

és més que un contenidor de dades que són requerides des de vàries classes. A la Figura 34 podem veure el disseny de classes resultant.

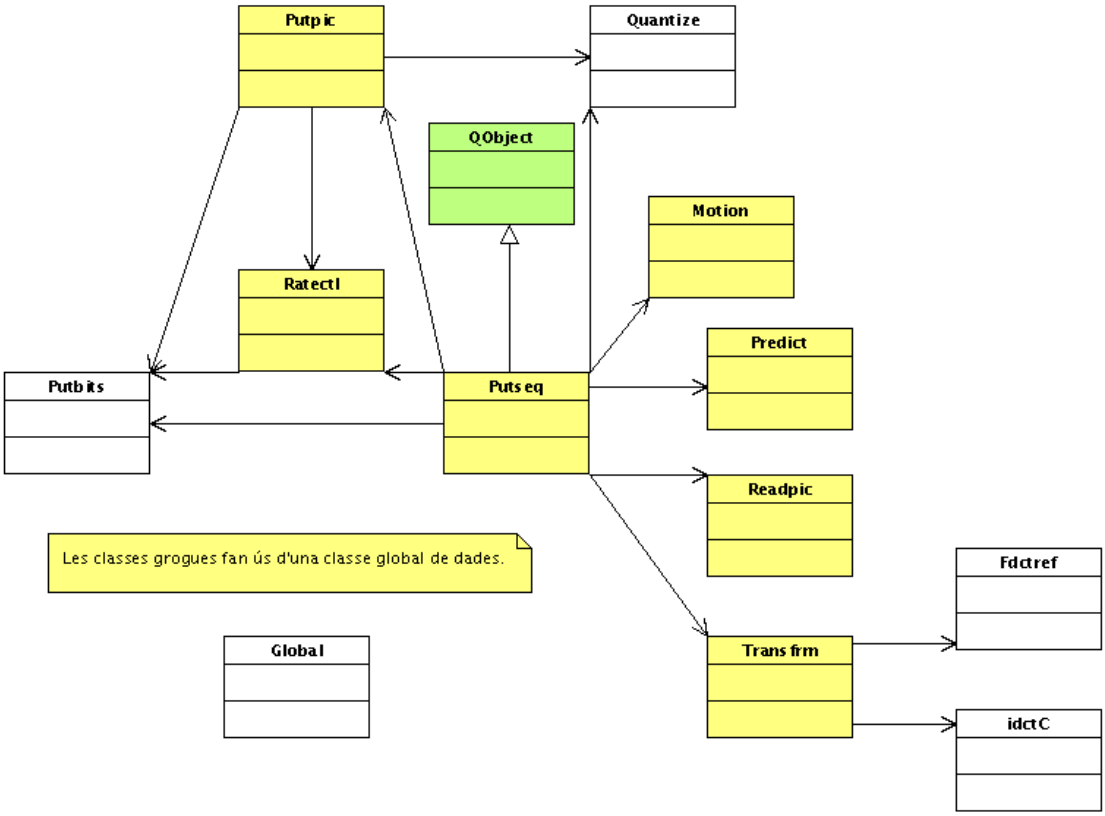


Figura 34: Disseny de classes del compressor

A grans trets el funcionament seria que la classe *Readpic* va llegint les imatges de memòria a partir de les quals es genera la seqüència de vídeo. Una de les modificacions que ha calgut fer en aquesta classe és que en funció dels paràmetres d'entrada se l'ha obligat a llegir més d'una vegada la mateixa imatge. Això és així perquè si tenim per exemple que volem reproduir 5 imatges a 5 imatges/segon, quan convertim el vídeo a format PAL, aquest necessàriament serà reproduït a 25 imatges/segon, de manera que el compressor haurà de comprimir 5 vegades cada imatge, per generar el segon de vídeo.
Les classes *Predict, Motion, Quantize* i *Ratectl* són per fer els càlculs que comprimeixen les imatges. I per escriure el fitxer resultant a disc s'utilitzen les classes *Putbits* i *Putpic.* Com que es produeix un canvi a la representació de les imatges s'utilitzen les classes *Fdctreg* i *idctC* per realitzar la transformada ràpida del cosinus. Una realitza la transformada directa i l'altra la transformada inversa.

La classe que ens serveix per accedir a les funcionalitats del compressor és la classe *Putseq.* Aquesta classe agafa els paràmetres necessaris de la classe *Global* i només cridant el mètode

putseq amb els paràmetres escollits per l'usuari genera el fitxer de vídeo a partir del conjunt de imatges seleccionades en el visor 2D. En cas de no haver seleccionat cap grup de imatges el vídeo es generarà a partir de totes les imatges.

Una altra modificació que ha calgut fer ha estat introduir un diàleg amb una barra d'estat que ens informa del percentatge de vídeo que portem generat. És fàcil que la generació d'uns pocs segons de vídeo duri minuts de manera que si l'aplicació no fes res l'usuari podria pensar que s'ha penjat.

Per actualitzar aquest diàleg de progrés s'ha fet heretar la classe *Putseq* de *QObject* i se li ha afegit un *signal* que és capturat per la classe pare.

S'ha pogut descartar la classe que validava que tots els paràmetres del fitxer de configuració fossin correctes. Aquesta classe s'ha suprimit ja que la majoria de paràmetres agafen valor per defecte i els paràmetres que escull l'usuari són validats abans de ser passats al compressor.

8.4 Paràmetres per defecte del compressor

Una opció que disposa el compressor és escollir la relació d'aspecte en què es mostrarà el vídeo. Els valors típics són 4/3 o 16/9, però com es veu a continuació, si la imatge original no té aquestes proporcions, la imatge es deforma, cosa que no interessa perquè podria semblar, per exemple, que el pacient té un cor més gran o aixafat del que en realitat és (vegis Figura 35). Afortunadament aquest valor es pot posar a 0 de manera que les imatges es mostrin amb les proporcions reals que tenen.

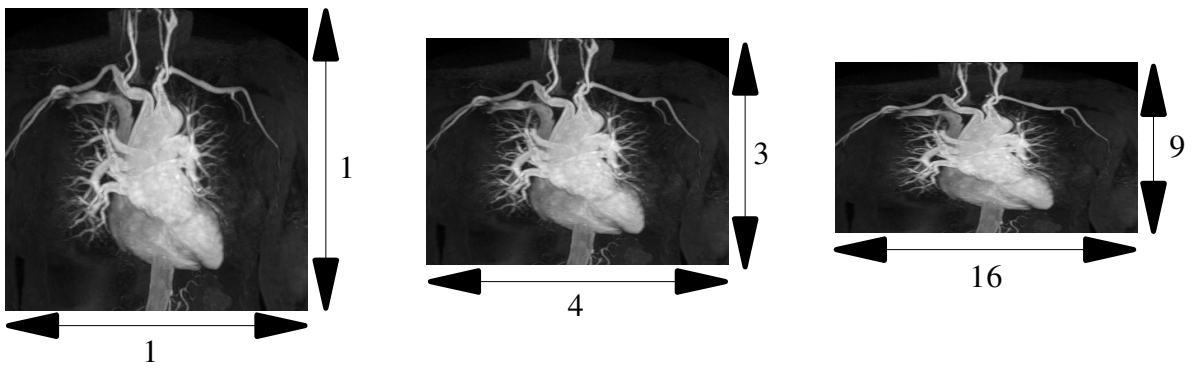


Figura 35: Diferents proporcions d'una mateixa imatge. D'esquerre a dreta, 1:1, 3:4, 16:9

Quan es genera vídeo en format similar al PAL o al NTSC, hi han una longitud típica per a l'estructura de GOP. Són 12 imatges pel vídeo PAL i 15 imatges pel vídeo NTSC. Són uns valors pels quals com a norma general el vídeo creat manté una bona qualitat de imatge i un bon nivell de compressió.

La distància entre els frames del GOP també és un valor que el compressor permet modificar. Podem escollir que no s'utilitzin imatges del tipus B o que s'utilitzin dels tres tipus possibles I, P i B. Vegis l'annex B per aclaracions.

8.5 Integració del compressor mpeg2 al Starviewer

L'exportació a format de vídeo s'ha afegit dins el Visor de imatges 2D. S'ha afegit aquí perquè pel context és on li pot interessar exportar el vídeo. La manera de funcionar és que l'usuari visualitzi les seqüències de vídeo des del Starviewer des d'on pot ajustar paràmetres com el contrast, la mida de les imatges, la velocitat de reproducció, etc. Si decideix que el vídeo que està veient el vol exportar ho pot fer clicant al botó de gravació.

8.5.1 Característiques del fitxer de vídeo exportat

El fitxer de vídeo que es genera té algunes característiques que el diferencien respecte el la reproducció de imatges que podem fer des del propi Starviewer, aquestes són:

- El fitxer de vídeo es genera a partir de les imatges contingudes al fitxer original i no en relació a les imatges mostrades a la pantalla. Això fa que la resolució de les imatges del vídeo tinguin la mateixa mida que les imatges presents al fitxer original.
- El contrast de la imatges no es correspon amb el que es veu a la pantalla. Això també és degut al fet de generar el vídeo a partir de les imatges originals i no de la imatge mostrada en pantalla. Aquesta característica serà detallada en el següent apartat.
- El que sí es conserva és la velocitat de reproducció. Si el vídeo que volem generar és a partir de mostrar les imatges a 5 frames/segon, tant si generem un vídeo a 25 frames/segon (PAL) com a 30 frames/segon (NTSC), les imatges es mostraran a la mateixa velocitat que l'usuari ha escollit al Starviewer.

8.5.2 Transformació a l'espai de color RGB

Per generar el fitxer resultant el compressor necessita treballar amb la representació YUV o RGB dels píxels, on cada píxel ocupa 3 bytes, un byte per component. En canvi, les imatges amb què es treballa en ressonància magnètica no tenen color i només consten d'un valor enter per píxel que guarda la lluminositat. Aquest valor no va de 0 a 255 sinó que depenent de la màquina els valors possibles es troben en un rang entre -1000 i +1000, o superior.

Per tant el primer pas abans d'enviar les imatges al compressor consistirà en representar la lluminositat a l'espai de color RGB o bé a l'espai YUV.

A partir dels coneixements adquirits sobre la formació de colors en assignatures com Visió per Computador, i sabent que els tons grisos a l'espai RGB es troben quan la component vermella (Red), com la verda (Green) i blava (Blue) tenen el mateix valor. Aprofitarem aquesta característica per fer que quan la lluminositat valgui 100, siguin en realitat la component R, G i B les que valguin 100. A la taula de la Figura 36 es vol mostrar com el valor de cada component influeix en el to de gris obtingut.

R	*G*	*B*	
0	0	0	
128	128	128	
255	255	255	

Figura 36: Taula amb diferents valors de gris en funció de cada component

Com es pot veure, quan més petits són els valors de cada component, més negre és el color, i com més grans són els valors, més blancs. Per valors grans de lluminositat, més tirant a blanc serà el color i com més petit sigui el valor, més proper a negre.
Per tant ja tenim que el primer pas consisteix en agafar el valor de lluminositat de cada píxel i utilitzar-lo per cada component del píxel.

8.5.3 Ajustament de l'escala de valors

L'altre problema que trobem és que els valors de lluminositat tenen un rang de valors superior als 256 valors que podem guardar en un píxel. Per tant caldrà que prèviament a utilitzar cada valor de lluminositat com a component de píxel canviem el rang de valors de manera que si la nostra imatge té un rang de valors que va de -1000 fins a 1000, l'haurem de convertir a un rang de valors entre 0 i 255 perquè el compressor funcioni.
Evidentment estem perdent informació, però és una limitació que tenim deguda a què el format mpeg2 està pensat per imatges en color o amb nivells de grisos de com a màxim 256 valors.
A la Figura 37 es veuen l'escala de valors de la imatge original i l'escala resultant un cop aplicada la transformació pertinent. De color negre a sobre de l'eix original es veu un franja més gruixuda que es correspon amb els colors que apareixen a una imatge tipus. Com es

pot veure no tots els 2000 colors apareixen a la imatge, sinó que només uns quants.

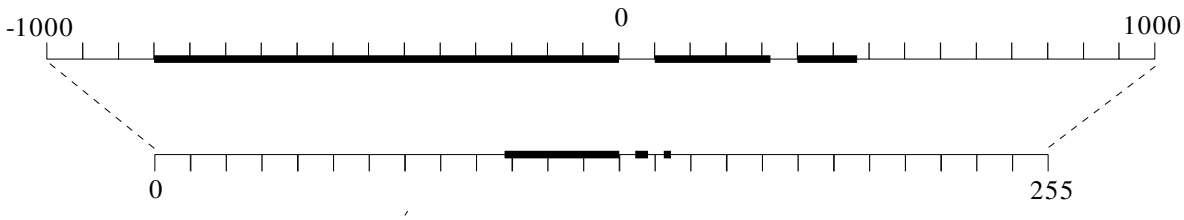


Figura 37: Conversió de píxels utilitzant el rang lògic de la imatge

La conversió realitzada en aquest exemple ha consistit en convertir els 2001 possibles valors als 256 que pot representar el vídeo mpeg2. Fixem-nos que fer la conversió d'aquesta forma té l'inconvenient que els valors es queden molt concentrats. Aquest fet provoca que la imatge tingui tots els píxels molt semblants entre sí, i defineixi una imatge poc contrastada.

Com que a les proves realitzades aplicant el canvi d'escala anterior les imatges apareixien pràcticament totes negres es va decidir aplicar un altre tipus de canvi d'escala. Per entendre'l ens fixarem que a l'exemple anterior hi ha valors molt grans i molt petits que no apareixen a la imatge. Són colors que la imatge no té. Per tant en comptes de convertir tot el rang de valors teòric de la imatge, podem convertir només el rang de valors que realment utilitza la imatge. Amb l'ajuda de la Figura 38 s'entén millor la idea.

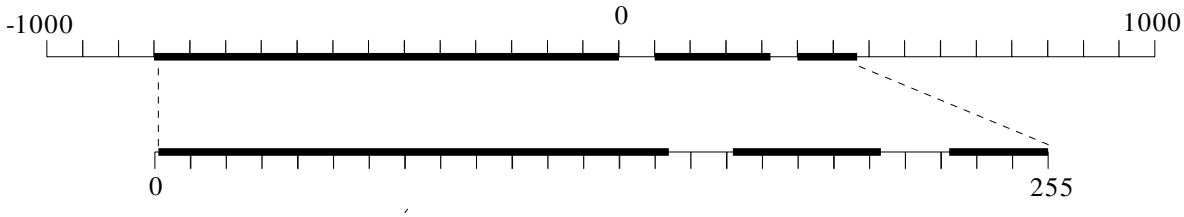


Figura 38: Conversió de píxels utilitzant el rang real de la imatge

La imatge anterior no ens ha d'enganyar i ha de quedar clar que la conversió implica pèrdua de informació, però la millor utilització del rang de valors que utilitzarem farà que les imatges tinguin un millor aspecte.

9 Visor 3D-4D per plans

La visualització de 3D per plans consisteix en la representació tridimensional d'un volum. El que es veu no és la reconstrucció sencera del volum, sinó 3 plans ubicats a cada eix ortogonal que es poden moure i mostrar què hi ha en la zona on s'ubica cada pla. Amb l'ajut de la Figura 39 s'entén millor l'explicació.

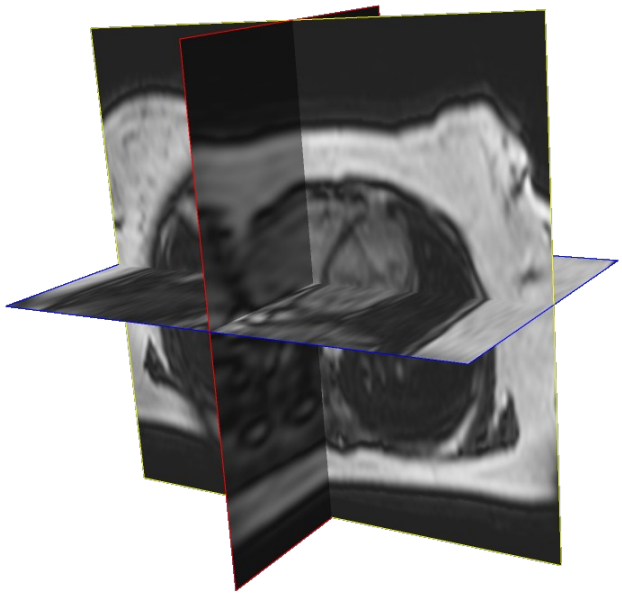


Figura 39: Visualització 3D d'un volum per plans

Els volums de què disposàvem no eren fitxers DICOM, sinó que eren en format *.mhd*. El format *mhd* és un format propi de les VTK i la informació de capçalera és molt breu, només inclou dimensions de les imatges i espai entre els píxels i les llesques.

Treballar amb imatges *mhd* va ser una necessitat i no pas una cosa pretesa. Com s'ha comentat anteriorment, tant l'exploració 3D com la 4D són proves que a dia d'avui no entren dins el protocol estàndard i obtenir més imatges requeria realitzar més ressonàncies magnètiques de manera que no va ser possible i ens vam haver de conformar amb les images que teníem.

El disseny d'aquest visor ha implicat crear noves classes i ampliar funcionalitats d'algunes ja existents. Les classes noves s'han afegit en el mòdul que hem anomenat *Gestió de visors* a la Figura 20 de l'apartat 5.6, i els mòduls que han vist ampliades i millorades les seves

són el mòdul *Entrada/Sortida* i el de *Gestió dels fitxers/volums oberts* de la mateixa figura.

9.1 Millores en el disseny

Les millores han consistit en la creació d'un visor que permetés veure tant un fitxer que contingués un únic model com un fitxer o directori que contingués varis models. A la Figura 40 podem veure quina és la relació entre les classes que forment aquest visor.

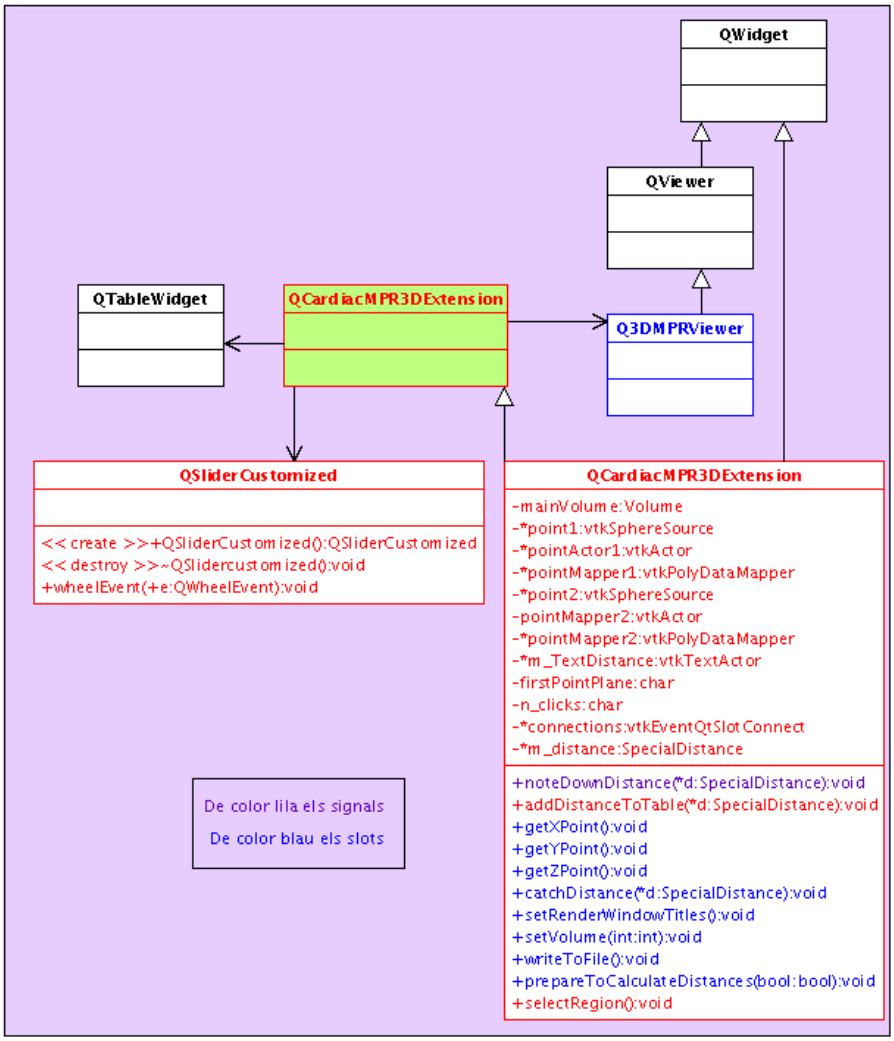


Figura 40: Classes implicades en la visualització 3D-4D per plans

A continuació es comenta en detall quina és la finalitat de cada classe que apareix al disseny de la Figura 40.

- **QSliderCustomized**

  Utilitzada també en el Visor 2D on s'ha comentat en detall. És útil perquè permet canviar la imatge que visualitzem fent girar la rodeta i avançant les imatges d'una en una.

- **QCardiacMPR3DExtensionBase**

  És la classe que integra els elements de la interfície com són bàsicament els botons que ofereixen les funcionalitats per reproduir les imatges, la classe Q3DMPRViewer que és l'encarregada de mostrar les imatges a l'àrea definida com a tal i l'element QTableWidget que és una taula que s'utilitza per emmagatzemar les distàncies prèviament calculades així com els valors necessaris per a poder tornar a visualitzar la disposició dels plans en l'instant en què s'ha calculat aquella distància.

- **QCardiacMPR3DExtension**

  És la responsable de calcular distàncies i anotar-les a la taula, així com de reproduir les seqüències de volums i permetre escollir una regió d'interès. Permet capturar l'escena i guardar-la en un fitxer .jpg.

- **Q3DMPRViewer**

  És l'encarregada de proporcionar els tres plans que es tracten de 3 vtkImagePlaneWidget així com de oferir les opcions relacionades amb ells. Com pot ser mostrar-los, ocultar-los o orientar el model en vista axial, sagital o coronal. També engloba l'actor de l'escena que permet dibuixar el cub que envolta el model.

Les classes QViewer és la classe de la qual deriva qual element que hagi de ser visible en una interfície d'usuari. Per aquest motiu tant la classe Qviewer com Qcardiac3DMPRExtension en deriven.

9.2 Ampliacions en el disseny

La incorporació d'aquest visor va ser bastant dificultosa pel fet de tractar dades per les quals no havia estat dissenyat. Es van haver de fer canvis a la classe Q3DMPRViewer per tal de permetre canviar el model un cop ja en teníem un de carregat, característica imprescindible per reproduir les imatges com si es tractés d'una pel·lícula. Però els canvis més important els trobem en la classe que guarda el fitxer carregat a memòria, la classe Volume que inicialment no havia estat pensada per contenir 4D.

A continuació s'expliquen aquests canvis que al ser força importants es mereixen especial atenció. Però abans no hem d'oblidar que la incorporació d'aquest visor també ha obligat a tocar les classes que el fan accessible des de l'aplicació principal. Aquesta classes són la QapplicationMainWindow que fa que aparegui com una nova entrada al menú visualitzation i la classe ExtensionHandler que associa què ha de passar quan l'usuari fagi

clic a la nova entrada del menú que hem afegit.

9.2.1 Càrrega d'un 4D a memòria

La classe Volume funcionava de manera que tot el que obria ho guardava com a volums de 3 dimensions Itk<Image> de manera que quan es necessitava que aquest volum fos visualitzat es veia la conversió a volum Vtk<Image> que sí podia ser mostrat pels visors que ofereixen les Vtk.

Com que els fitxers 4D amb els quals treballaven no eren DICOM, aquest ja tenien les capçaleres del fitxers retocades de manera que el que deien és que aquell fitxer en qüestió contenia 3D. Amb aquesta capçalera quan es carrega el fitxer a l'aplicació s'interpreta la tercera dimensió com totes les imatges que s'han capturat al llarg del temps i no només les referents a un instant.

La primera intenció va ser voler modificar el disseny de la classe Volume de manera que es poguessin llegir models de 4 dimensions correctament, sense necessitat de fer aquesta petita conversió per la qual es feia passar un 4D per un 3D amb tantes llesques com instants de temps per llesques d'un volum s'havien capturat.

Es va analitzar quines classes es veien afectades i es va parlar amb els responsables del manteniment del Starviewer. El consell va ser que si es podia fer sense haver de fer aquest canvi millor. Tot i ser una feina complexa dotar un disseny de funcionalitats que inicialment no s'han previst, em va semblar lògic que si aquest projecte només havia de ser una ampliació de la feina ja existent proposar un canvi tant important a nivell de disseny, no era feina d'un projectista decidir si el canvi era bo i per tant vaig cenyir-me al consell.

Tot i que segurament la millor opció segurament seria el redisseny de la classe Volume és comprensible que abans de fer aquest pas els desenvolupadors del Starviewer s'ho vulguin pensar ja que no hem d'oblidar que les captures 4D del cor a dia d'avui només es fan com a experiment.

Com que com ja hem comentat les capçaleres del fitxers amb el quals tractàvem havien estat modificades per poder ser oberts per el Starviewer, el que es va decidir és que per defecte qualsevol fitxer s'obrís amb el visor 2D i llavors l'usuari, si veia que la visualització que tenia a davant es tractava d'un 4D, que introduís el nombre de models que contenia

aquell fitxer i escollís utilitzar el visor per plans.

El disseny final de la classe Volume el podem veure a la Figura 41 on s'ha ressaltat amb negreta els mètodes i atributs que ha calgut afegir.

Figura 41: Classe Volume resultant on es pot veure en negreta els mètodes afegits

Amb la classe Volume que s'ha dissenyat és possible carregar un fitxer que contingui un 4D i visualitzar només un model. Per fer-ho el que es fa és carregar el fitxer com si d'una Itk<Image> es tractés, però un cop l'usuari informa a la classe que es tracta d'un 4D, la classe Volume monta una llista de Vtk<Image> separant cada model amb les seves llesques corresponents.

S'ha fet que el redisseny no afecti a les classes ja dissenyades fent que independent de si la classe treballa amb la llista de volums o amb un únic volum les consultes de fora tinguin sentit i es facin a qui toca.

Un dels problemes que vam tenir i que ens va obligar a fer més canvis va ser que per construir els diferents volums es va construir un pipeline que anava extraient els volums de la classe itkData per transformar-los a vtkData. Doncs bé, el pipeline no suporta que un cop construït li canviïs l'entrada i en el nostre cas l'entrada s'havia d'anar variant ja que per cada volum que volíem construir l'entrada era un tros diferent de la imatge itk. Per solucionar-ho s'havia de construir el pipeline a cada iteració.

Un cop solucionat tot els entrebancs per separar els diferents models, va resultar que els fitxers 4D no estaven capturats per models, sinó per llesques. Primer trobem la llesca 1 capturada a l'instant 0, a l'instant 1, a l'instant 2, i així fins a l'instant N-1. A continuació ve la llesca 2 capturada durant N instants més. I així fins tenir les captures de les M llesques que formen el model.

El que a nosaltres ens interessava era que un cop carregat el fitxer a memòria, les primeres M llesques es corresponguin al primer model, les següent M llesques al següent model, i així fins tenir els N models que podem reconstruir a partir de la informació que ens aporta aquell fitxer. Per tant si volíem reconstruir els diferents models calia ordenar les llesques.

9.2.1.1 Ordenació del model

Ordenar el model com és evident consisteix en canviar les llesques de lloc, però hem de tenir present que moure imatges de lloc encara que sigui a memòria és una tasca lenta i més si tenim en compte que estem treballant amb 4D que fàcilment ocupen els 200 MB.

Per ordenar les llesques podríem reservar un espai de memòria temporal per tal de guardar la llesques que són sobreescrites abans de ser mogudes a la posició que realment els correspon i d'aquesta manera quan calgués moure una llesca, l'hauríem de buscar primerament a l'espai de memòria temporal i en cas de no trobar-la llavors l'hauríem de buscar dins el propi model 3D.

Aquesta primera forma de reordenar les llesques presenta bàsicament dos problemes:

- Utilitza molta memòria ja que requereix de un buffer on guardar les llesques que han estat substituïdes a més a més d'un control per saber quina era la seva posició.
- I l'altre problema és que és lent ja que hi ha llesques que abans de ser mogudes a la posició que els pertoca han de passar pel buffer i per tant el cost de moure aquella llesca es veu multiplicat per dos pel fet d'haver-la hagut de moure dues vegades dins la memòria.

Aquests dos problemes ens van fer pensar si era possible moure les llesques en un determinat ordre de manera que no fos necessari moure tantes vegades les llesques abans de trobar la seva posició. I la resposta va ser que sí. Per trobar el mètode es va fer de manera empírica a partir de varis exemples. Anem a veure com es va arribar a la solució final a partir d'un exemple inventat on es treballarà amb un nombre petit de llesques per facilitar-ne la comprensió.

Suposem que tenim un fitxer que conté 12 llesques on cada llesca ha estat capturada en 3 instants de temps diferents. Per tant podrem reconstruir el mateix model en 3 estats de temps diferents. Com podem esbrinar de fer la divisió 12 / 3, tenim que cada model està format per 4 llesques. A la taula següent es mostra com queda col·locada cada llesca de l'exemple.

Llesca	***Disposició inicial (per temps)***	***Disposició final (per volums)***	***Volum***
0	0	0	**0**
	1	3	
	2	6	
1	3	9	
	4	1	
	5	4	**1**
2	6	7	
	7	10	
	8	2	**2**
3	9	5	
	10	8	
	11	11	

Com podem veure la primera llesca i la última no canvien, i la resta en podem definir la seva posició a partir de la següent fórmula:

I = Instants de temps, *4 a l'exemple*

M = Models, *3 a l'exemple*

$Y(s) = (s \ mod \ I) * M + (s \ div \ I)$

Sabent la fórmula que ens diu quina llesca va a la posició per la qual preguntem, anem a veure les passes que segueix l'algorisme fins tenir el model ordenat:

Mentres no *totes_llesques_ordenades* **fer**

GuardarABuffer(*1a_llesca_no_ordenada*)

llesca_a_moure := Y(Posició(*1a_llesca_no_ordenada*))

Mentres *llesca_a_moure* ≠ Posició(*1a_llesca_no_ordenada*) **fer**

CopiarOrigenDesti (Imatge(*llesca_a_moure*), Imatge(Posició(*1a_llesca_no_ordenada*)))

1a_llesca_no_ordenada = *llesca_a_moure*

Fmentres

CopiarOrigenDesti (GetImatgeDeBuffer, Imatge(Posició(1a_llesca_no_ordenada)))

BuscarLlescaDesordenada()

Fmentres

Una explicació més informal del funcionament de l'algorisme és la següent. Es mou una llesca a la seva posició destí i es col·locar a la posició que queda buida la llesca que li pertoca, es repeteix el procés amb l'espai que deixa lliure aquesta aquesta segona llesca que s'ha mogut. I així fins que haguem mogut totes les llesques al seu lloc.

9.3 Funcionalitats del visor 3D-4D per plans

Amb el Visor 3D-4D per plans es poden dur a terme les següents tasques:

- Visualitzar un model 3D amb l'ajuda de tres plans ortogonals que es poden moure i mostren en la seva superfície la zona del model que tallen.
- És possible ajustar el contrast de les imatges mostrades pels plans.
- Consultar les coordenades d'un punt així com el seu valor d'intensitat.
- Donar una orientació determinada als plans i reproduir el model en cas que es tracti d'un 4D. Igual que en el visor 3D també es permet reproduir un rang de imatges que pot ser escollit amb l'ajuda de la barreta de desplaçament (slider) i els botons per a tal propòsit. Es permet controlar la velocitat de reproducció i reproduir cíclicament les imatges.
- Es permet col·locar el model amb una de les posicions preestablertes amb un simple clic. Les posicions possibles són vista axial, sagital i coronal.
- Es poden suprimir els plans que molestin per fer una visualització determinada.
- Es pot posar i treure el cub embolcant que inclou el volum al seu interior.
- Es poden calcular distàncies marcant els punts seleccionats a sobre dels plans amb un simple clic de ratolí. Els valors es van anotant a una taula lateral que associa a

cada pla un color de manera que cada mesura introduïda serà d'un color en funció del pla al qual correspongui.

- Es pot seleccionar una zona d'interès del volum a través dels plans de manera que es descarta la resta del volum.

10 Visor 3D

El visor 3D és la última funcionalitat afegida a l'aplicació i ha estat un a més a més en aquest projecte ja que els altres visors han ocupat la major part del temps i amb el visor 3D multiplanar es donava per assolit l'objectiu marcat com a visualització 3D. El que s'ha fet ha estat aprofitar que es disposava d'una classe que ens permetia visualitzar elements 3D mitjançant algunes de les tres tècniques següents:

- Ray Casting

  La visualització utilitzant algorismes de ray-casting consisteix en una intersecció observador - píxel - model. Per a cada píxel de la pantalla en què es vol visualitzar el model, es llença un raig des de l'observador i cap al píxel en qüestió (veure Figura 42). De tots els punts del model que travessa el raig llençat, ens quedem amb el més proper a l'observador i es calcula la il·luminació d'aquest punt per poder-lo pintar del color adequat al píxel corresponent. En aquest cas per pintar el píxel s'utlitza la composició de color.

- Iso Surface

  És una tècnica que permet reconstruir la superfície de l'objecte sense deixar-ne veure el contingut.

- Projecció de màxima intensitat (MIP)

  És la mateixa idea que els algorismes de Ray Casting però per renderitzar s'utilitza el valor de màxima intensitat.

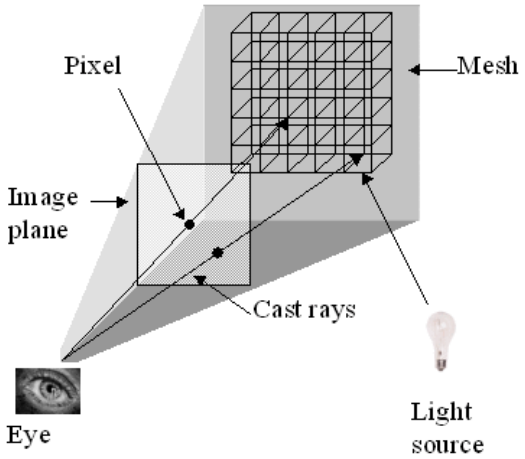


Figura 42: Explicació del Ray Casting

El que s'ha fet és dotar a l'usuari d'una interfície per visualitzar 3D i escollir l'algorisme a utilitzar. Les funcions de transferència estan preprogramades i part de la feina ha consistit en mirar d'ajustar-ne els valors per tals de representar la zona del cor el millor possible.

A la Figura 43 ressaltat amb vermell es mostren les classes que s'han hagut de crear. La classe de color verd igual que al llarg de tota aquesta memòria es correspon a una interfície d'usuari. La classe QCardiac3DViewer incorpora mètodes per tal de carregar un volum al visor i que al desplaçar el slider aquest canviï el volum mostrat per l'anterior o el posterior en funció del sentit.

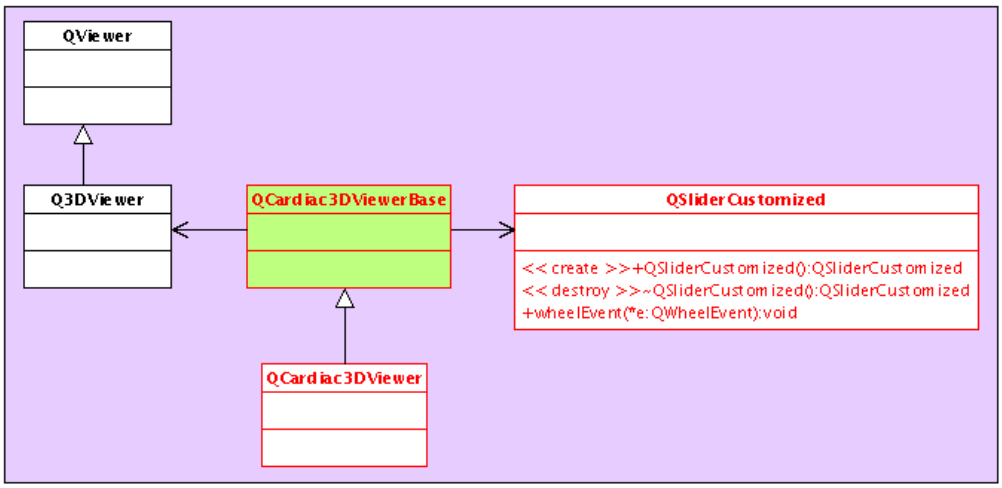


Figura 43: Diagrama de classes del visor 3D

Les proves realitzades amb aquest tipus de visualització han fet palès que calia millorar-ne el comportament ja que la reproducció de 4D era impossible a causa que l'ordinador no donava a l'abast.

11 Ampliacions i millores

Aquest projecte ha estat una primera incursió a l'anàlisi de dades de cor per part del Starviewer. Les possibles millores a afegir en un futur poden ser moltes, però en destacarem algunes:

- Fer més flexible l'exportació a vídeo permetent generar fitxers de vídeo a partir de models 4D, a partir de les accions que realitza un usuari sobre el model 3D. O bé poder generar el fitxer a partir de la zona seleccionada de la pantalla.
- Dotar l'aplicació de més eines de consulta que puguin se urilitzades per l'usuari. Per exemple poder calcular la intensitat d'una mateixa zona en diferents instants de temps serviria al metge per calcular la velocitat a la qual circula la sang per aquell punt.
- Fer que la interacció amb el visor 3D per plans permeti fixar el pla i moure el model, a més a més de com ho fa ara que fixa el model i permet moure el pla. Tot i que ara els resultats obtinguts són bons té el petit inconvenient que si allunyem el pla per veure un zona posterior del volum, el pla s'ubica en una altra àrea de la pantalla fet que dificulta que puguem fixar la vista amb el seu contingut.
- Permetre seleccionar regions en la representació 3D. Seria útil perquè podríem descartar les parets del tòrax per poder-nos centrar millor en el cor. A més a és s'haurien de poder variar les funcions de transferència de la interfície d'usuari per poder ajustar els valors correctes mentre es visualitza el model.
- Buscar les funcions de transferència de color que millor representin el cor i les venes del voltant.
- Donar la opció de guardar els models 4D un cop carregats. Amb aquesta funcionalitat es buscaria que si s'ha de treballar molt sovint amb un mateix model que té les llesques desordenades, la operació d'ordenació només fos necessària fer-la una vegada. Llavors podríem carregar i treballar a partir del model ordenat.

12 Conclusions

L'objectiu d'aquest projecte era desenvolupar i integrar en una plataforma de processament de dades mèdiques els mòduls necessaris per poder tractar, manipular i visualitzar dades cardíaques.

De manera que s'han assolit tot els objectius proposats, ja que l'aplicació desenvolupada permet:

1. Visualització imatges 2D
2. Visualització 3D del volums adquirits
3. Visualització de models 4D
4. Esportar les dades a formats diferents dels que proporciona originàriament el dispositiu de captació permetent per exemple visualitzacions en format vídeo.
5. Visualització i manipulació per zones d'interès (selecció de ROI's)
6. Interrogació del model si bé només s'ha dotat de la capacitat de mesurar distàncies.

Les dades han pogut ser validades a partir de casos resultats donant resultats satisfactoris.

12.1 Conclusions personals

Un cop acabat el projecte podem dir que els objectius proposats inicialment s'han assolit. Ha estat un projecte que com més anava avançant el seu desenvolupament, més engrescador es tornava perquè les coses anaven sortint amb més facilitat. Començar va ser difícil perquè es tractava de millorar un projecte relativament gran i avançat, la qual cosa imposava una sèrie de restriccions (o guies depenent de com es miri) a l'hora d'afegir noves funcionalitats.

El fet de utilitzar tres llibreries extenses i complexes com són les QT, ITK i VTK en especial aquestes dues anteriors, va fer que hagués de dedicar molt de temps a la fase d'aprenentatge. La incursió en el món de la visualització 3D també ha estat nova per mi ja que anteriorment mai havia cursat cap assignatura relacionada. Per tot plegat considero que la realització d'aquest projecte ha estat una experiència molt positiva pel fet d'haver-me introduït a remenar una temàtica i un seguit d'eines que al no haver-hi treballat mai m'han servit per aprendre i obrir nous horitzons. També m'agradaria dir que per mi ha estat molt positiu poder anar a l'hospital Josep Trueta i poder saber de primera mà quines

funcionalitats busca un expert que tracti amb ressonàncies de cor, en una aplicació d'aquestes característiques.

També valoro molt positivament es suport rebut per part dels desenvolupadors del Starviewer, que gràcies a ells les dificultats es feien més petites.

13 Manual d'usuari

En aquest capítol aprofitarem per fer un manual d'usuari i al mateix temps ensenyar quins són els resultats que s'obtenen de visualitzar les imatges de cor utilitzant aquesta aplicació

13.1 Visor 2D

L'aspecte que presenta aquest visor el podem veure a la Figura 44 i a continuació podem veure com utilitzar les funcionalitats que disposa.

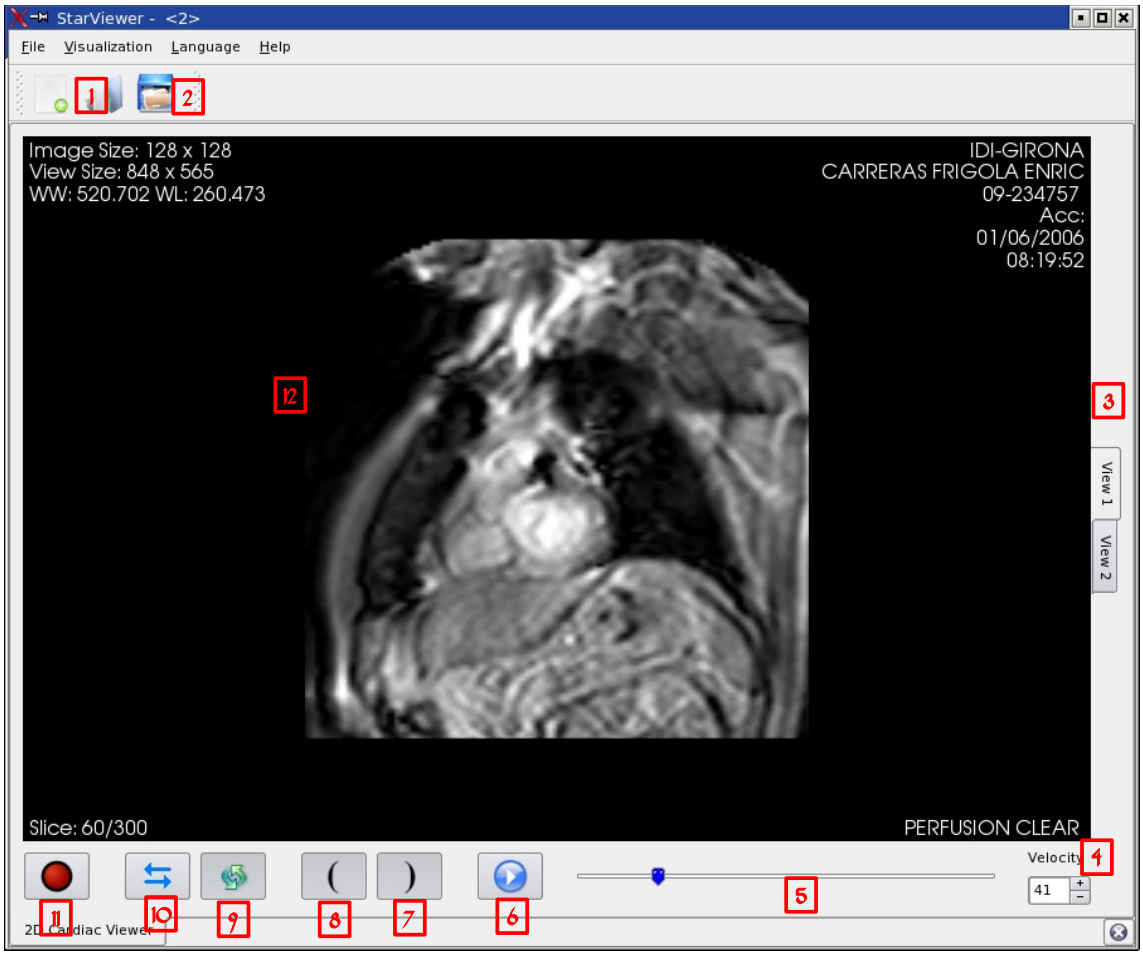


Figura 44: Visor 2D mostrant unes proves de perfusió

1.- Obrir un fitxer. També podem anar al menú *File / Open...*

2.- Obrir un directori. També podem anar al menú *File / Open DICOM directory.*

3.- Mostra tantes pestanyes com captures de diferents punt de vista continguin els fitxers que es mostren.

4.- Permet controlar la velocitat de la reproducció d'una seqüència de imatges. Les unitats són imatges per segon i tant ho podem fer amb la rodeta del ratolí com amb el botons més i menys del widget.

5.- Barra que permet desplaçar-nos per totes les imatges que contingui el fitxer o el directori que hem obert. Amb la rodeta del ratolí es poden saltar les imatges d'un en un. L'indicador de la barreta es torna blau quan es selecciona un rang de imatges.

6.- Permet reproduir o parar la reproducció de les imatges a la velocitat indicada a la part inferior dreta de la pantalla.

7.- Limita quina serà l'última imatge que es reproduirà. El botó es manté premut mentre l'opció està activada. També es evidencia que l'opció està activada que l'indicador del slider es torna de color blau. Per escollir la darrera imatge en hi posicionem utilitzant la barra de desplaçament i cliquem al botó.

8.- Limita o indica quina serà la primera imatge que es reproduirà. Per utilitzar-lo cal posicionar-nos a la imatge que volguem i prémer el botó. Per posicionar-nos utilitzarem al barreta de desplaçament.

9.- Si es prem la reproducció es produirà de manera cíclica de manera que quan s'arribi a l'última imatge es començarà a partir de la primera.

10.-Si es prem la reproducció serà repetitiva de manera que quan s'arribi al final es reproduiran les imatges al revés fins arribar a la primera que es tornaran a reproduir cap endavant, i així fins que no desmarquem el botó o tornem a prémer play.

11.-Botó que dóna accés a la exportació de vídeo.

12.-Àrea on apareix la imatge que es visualitza. A través dels botons del ratolí es pot variar el contrast de la imatge, fer-li zoom o traslladar-la de posició.

13.2 Exportació a format de vídeo

Quan es clica al l'exportació de vídeo clicant damunt la icona etiquetada amb el número 11 a la Figura 44 apareix el següent diàleg a pantalla. Les explicacions són prou clares i el que permeten escollir és el nombre de imatges per segon que s'utilitzaran per codificar el vídeo.

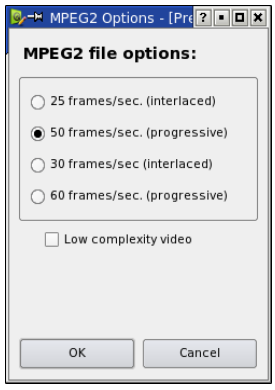


Figura 45: Diàleg amb les opcions disponibles per generar mpeg2

Per generar vídeo de baix cost computacional cal marcar la opció "Low complexity video". Com que la generació de vídeos de pocs segons és un procés llarg l'aplicació mostra un diàleg amb el percentatge que es porta completat de manera que l'usuari no tingui la sensació que l'ordinador s'ha penjat.

13.3 Visor 3D-4D per plans

L'aspecte que presenta aquest visor "dos en un" el tenim a la Figura 48, on podem veure un model en vista axial. Com que la vista axial és perpendicular a la resta de plans, aquests només es veuen de color blau i color vermell creuant la imatge i formant una creu al centre.

Per accedir a aquest visor primer cal obrir el model amb els botons 1 o 2 de la Figura 48 i un cop l'usuari hagi comprovat que es tracta d'un model 3D o 4D ha d'accedir al menú *Visualization* i escollir l'opció *3D-4D Cardiac Viewer Planes*. Primerament se li preguntarà si el model està ordenat o cal ordenar-lo.

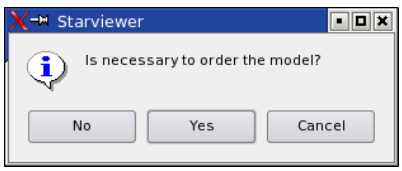


Figura 46: Diàleg on es demana si s'ha d'ordenar el model

En cas que calgui ordenar el model es demana quants models conté el fitxer. Per fer-ho el que pregunta és quantes llesques es veuen semblants fins que n'apareix una de molt diferent. El canvi es produeix quan apareix una llesca que es correspon a un altra zona del tòrax normalment.

Figura 47: Diàleg on es demana el nombre de volums que conté el fitxer

Mentre s'ordenen les llesques l'aplicació utilitza un diàleg de progrés per informar a l'usuari del percentatge de llesques ordenat.

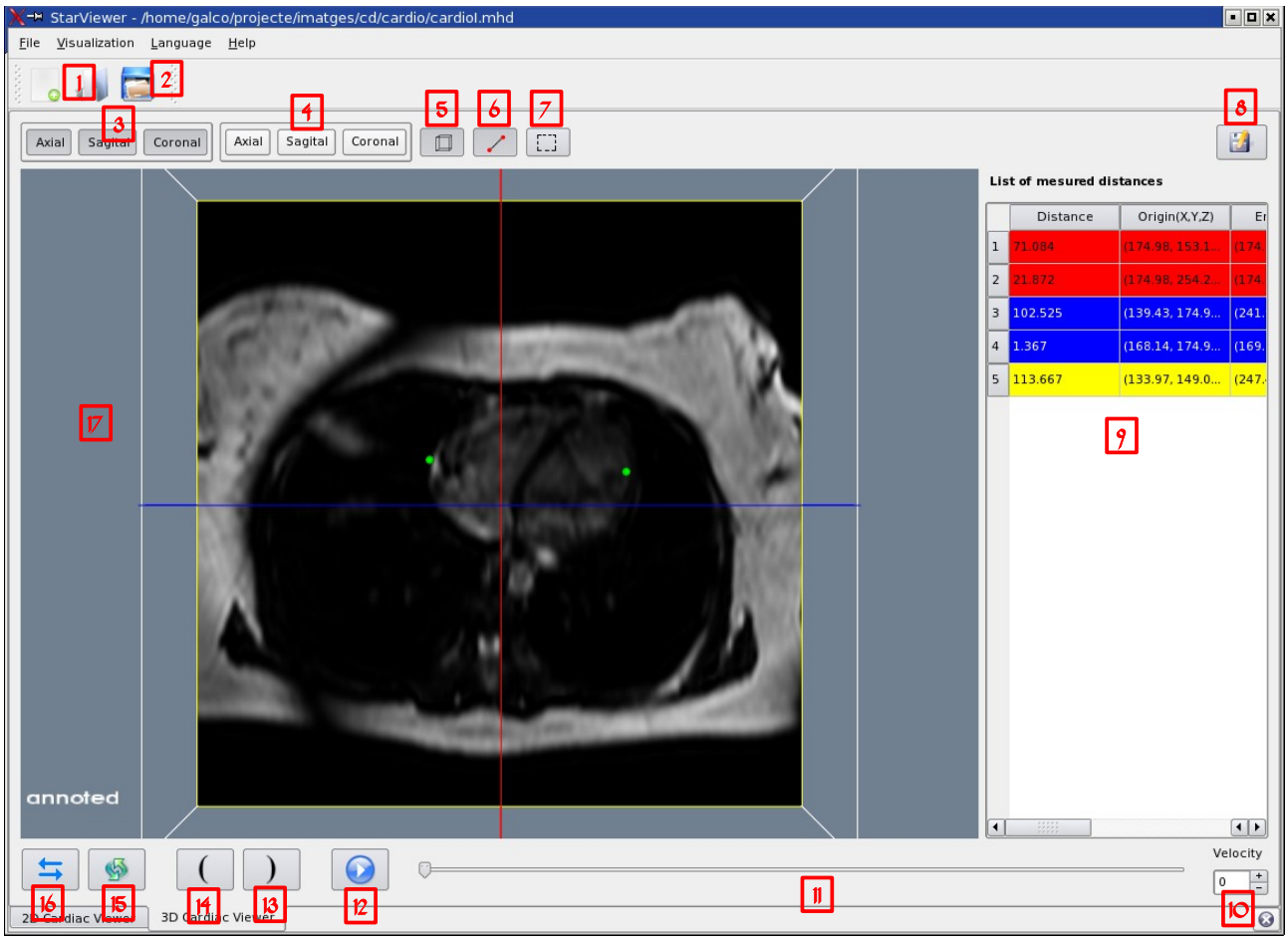


Figura 48: Visor 3D-4D

1.- Obrir un fitxer. També podem anar al menú File / Open...

2.- Obrir un directori. També podem anar al menú File / Open DICOM directory.

3.- Permet seleccionar quins plans volem mostrar al model

4.- Permet moure el model i mirar-lo des de 3 punts de vista diferents. Vista axial, sagital i coronal.

5.- Activa o desactiva el cub blanc que envolta el model.

6.- Eina que permet mesurar distàncies clicant als dos punts del pla entre els quals volem mesurar-ne la distància.

7.- Eina que permet escollir quina zona de la pantalla centrarà el nostre interès descartant la resta.

8.- Obre un diàleg que permet guardar el model que es veu a la pantalla en format jpg.

9.- Llista on van apareguent les distàncies que mesura l'usuari. El color del fons de l'entrada es correspon amb el pla sobre el qual s'ha fet la mesura. Clicant-hi al damunt disposa els plans exactament en la mateixa posició que quan s'ha mesurat aquella distància. Al mateix temps mostra de color verd quins són els punt mesurats.

10.-Permet controlar la velocitat de la reproducció d'una seqüència de imatges. Les unitats són imatges per segon i tant ho podem fer amb la rodeta del ratolí com amb el botons més i menys del widget.

11.-Barra que permet desplaçar-nos per totes les imatges que contingui el fitxer o el directori que hem obert. Amb la rodeta del ratolí es poden saltar les imatges d'un en un. L'indicador de la barreta es torna blau quan es selecciona un rang de imatges.

12.-Permet reproduir o parar la reproducció de les imatges a la velocitat indicada a la part inferior dreta de la pantalla.

13.-Limita quina serà la última imatge que es reproduirà. El botó es manté premut mentre l'opció està activada. També es evidencia que l'opció està activada que l'indicador del slider es torna de color blau. Per escollir la darrera imatge en hi posicionem utilitzant la barra de desplaçament i cliquem al botó.

14.-Limita o indica quina serà la primera imatge que es reproduirà. Per utilitzar-lo cal posicionar-nos a la imatge que volguem i prémer el botó. Per posicionar-nos utilitzarem al barreta de desplaçament.

15.-Si es prem la reproducció es produirà de manera cíclica de manera que quan s'arribi a l'última imatge es començarà a partir de la primera.

16.-Si es prem la reproducció serà repetitiva de manera que quan s'arribi al final es reproduiran les imatges al revés fins arribar a la primera que es tornaran a reproduir cap endavant, i així fins que no desmarquem el botó o tornem a prémer play.

17.-És la zona on es mostren els plans que permeten visualitzar el volum. Clicant damunt dels plans ensenya quina és la posició dels punt on hem clicat i quin és el seu valor d'intensitat.

El plans que tallen el model i alhora ens permeten visualitzar-lo tenen un comportament especial en funció de la zona on haguem clicat amb el botó central del ratolí.

A la Figura 49 es mostren quines són les zones que permeten cada comportament. Clicant a les arestes podem fer girar el pla en el sentit que desitgem. Clicant als laterals i movent el ratolí inclinem el pla endavant i endarrere respecte un eix imaginari fixat al mig del pla i paral·lel a la direcció del costat on hem clicat. I finalment l'últim moviment possible que ens permet el pla és moure'l en profunditat endavant i endarrere sobre la seva normal.

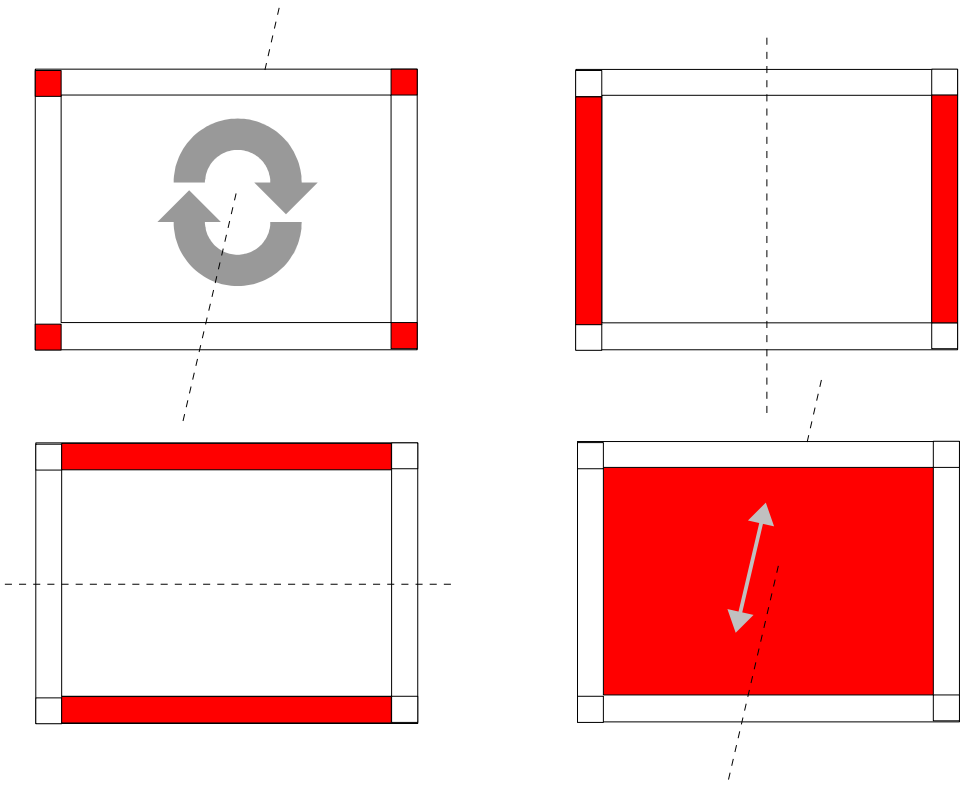


Figura 49: Zones clicables dels plans del visor per plans

13.4 Visor 3D

L'aspecte que presenta el visor 3D és d'allò més simple amb els elements mínims per portar a terme la tasca per la qual ha estat pensat. (Vegis Figura 50).

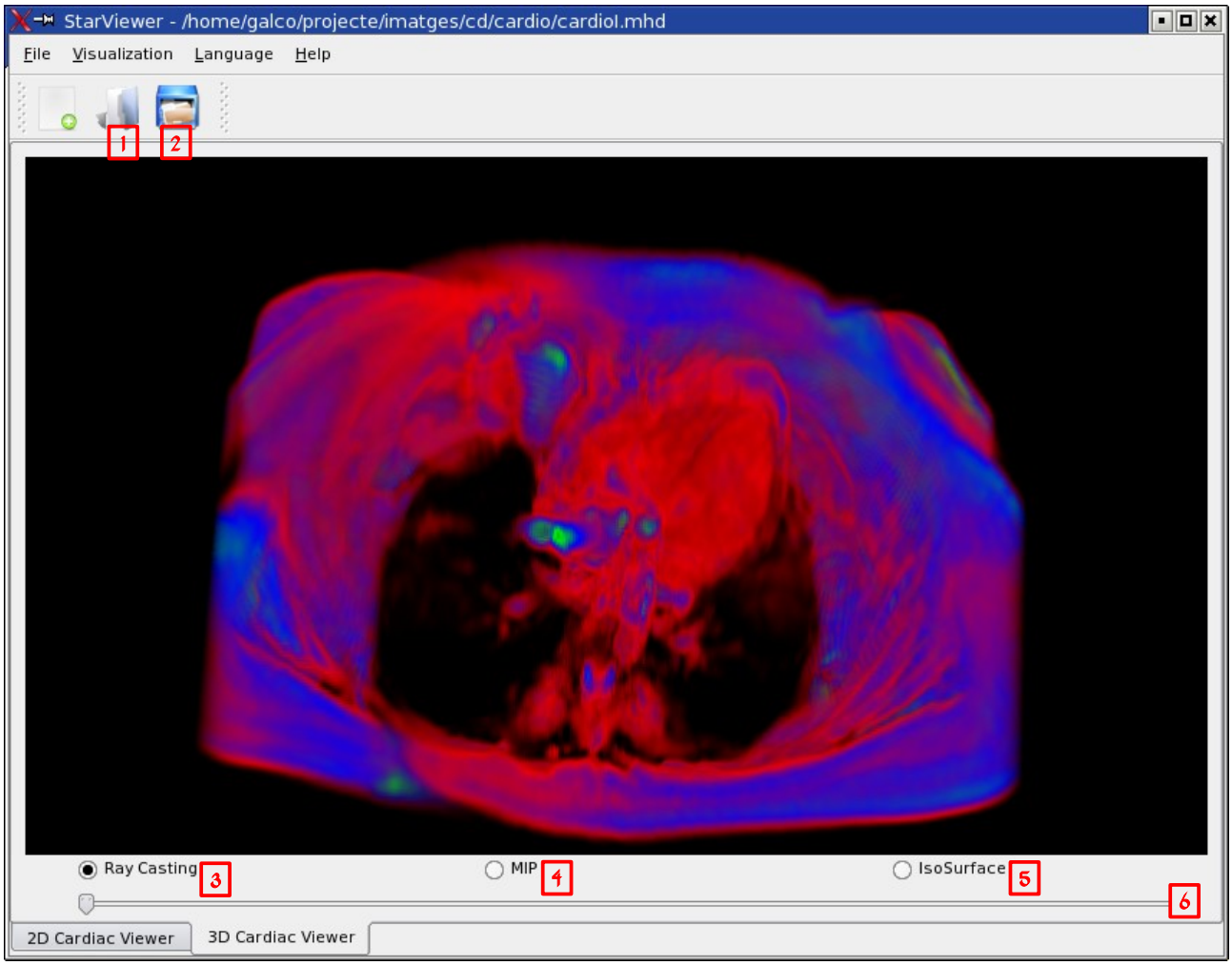


Figura 50: Vista 3D del cor utilitzant Ray Casting

1.- Obrir un fitxer. També podem anar al menú File / Open...

2.- Obrir un directori. També podem anar al menú File / Open DICOM directory.

3.- Mostra el volum aplicant la tècnica de Ray Casting.

4.- Mostra el volum aplicant la tècnica de MIP.

5.- Mostra el volum aplicant la tècnica de reconstrucció de superfícies Iso Surface.

6.- Barra de desplaçament per visualitzar els volums oberts quan es tracta d'un 4D.

ANNEX A – Capçalera d'un fitxer DICOM

Format:	DCM (Digital Imaging and Communications in Medicine image)
Class:	PseudoClass
Geometry:	256x256
Type:	Grayscale
Endianess:	Undefined
Colorspace:	Gray
Channel depth:	
Gray:	16-bits
Channel statistics:	
Gray:	
Min:	16 (0.000244144)
Max:	19268 (0.294011)
Mean:	4626.97 (0.070603)
Standard deviation:	3970.93 (0.0605925)
Colors:	65536
Rendering intent:	Undefined
Resolution:	72x72
Units:	Undefined
Filesize:	135kb
Interlace:	None
Background color:	white
Border color:	#DFDFDFDFDFDF
Matte color:	grey74
Page geometry:	256x256+0+0
Dispose:	Undefined
Iterations:	0
Compression:	Undefined
Orientation:	Undefined
Dcm::	normalized
Dcm:AcquisitionDate:	20060601
Dcm:AcquisitionNumber:	7
Dcm:AcquisitionTime:	101522.18
Dcm:DeviceSerialNumber:	05479
Dcm:EchoNumber(s):	1
Dcm:EchoTime:	1.48759996891021
Dcm:EchoTrainLength:	0
Dcm:FlipAngle:	40.0
Dcm:FrameofReferenceUID:	1.3.46.670589.11.0.0.11.4.2.0.5479.5.5712.2006060109470728000

Dcm:HeartRate:	0
Dcm:HighR-RValue:	0
Dcm:ImageDate:	20060601
Dcm:ImagedNucleus:	1H
Dcm:ImageOrientation(Patient):	0.999952733516690009721520356831076580228985.400001107351668179-.9999386668205
Dcm:ImagePosition(Patient):	-217.13931745290-1.591257830150283845252735333
Dcm:ImageTime:	101522.18
Dcm:ImageType:	ORIGINALPIMARYMFFEMFFE
Dcm:ImagingFrequency:	63.8973979999999
Dcm:Instance(formerlyImage)Number:	3
Dcm:InstanceCreationDate:	20060601
Dcm:InstanceCreationTime:	110427
Dcm:InstanceCreatorUID:	1.3.46.670589.11.5479.5
Dcm:InstitutionalDepartmentName:	IDI-GIRONA
Dcm:InstitutionName:	IDI-GIRONA
Dcm:IntervalsAcquired:	0
Dcm:IntervalsRejected:	0
Dcm:LowR-RValue:	0
Dcm:MagneticFieldStrength:	1.5
Dcm:Manufacturer:	Philips Medical Systems
Dcm:Manufacturer'sModelName:	Intera
Dcm:Modality:	MR
Dcm:NumberofAverages:	1.0
Dcm:NumberofPhaseEncodingSteps:	128
Dcm:NumberofTemporalPositions:	130
Dcm:Patient'sBirthDate:	19270408
Dcm:Patient'sID:	09-306708
Dcm:Patient'sName:	RIUS POUS DOLORS
Dcm:Patient'sSex:	F
Dcm:Patient'sWeight:	50.0
Dcm:PatientPosition:	HFS
Dcm:PercentPhaseFieldofView:	100.0
Dcm:PercentSampling:	50.0
Dcm:PhaseEncodingDirection:	ROW
Dcm:ProtocolName:	BolusTrak
Dcm:ReceivingCoil:	Q-Body
Dcm:ReconstructionDiameter:	450.0
Dcm:ReferencedImageSequence:	þÿ
Dcm:ReferencedStudyComponentSequence:	þÿ

Dcm:RepetitionTime:	5.45620012283325
Dcm:ScanningSequence:	GR
Dcm:ScanOptions:	PFF
Dcm:SequenceVariant:	SP
Dcm:SeriesDate:	20060601
Dcm:SeriesDescription:	BolusTrak
Dcm:SeriesInstanceUID:	1.3.46.670589.11.0.0.11.4.2.0.5479.5.3792.2006060110152081252
Dcm:SeriesNumber:	701
Dcm:SeriesTime:	101520.79000
Dcm:SliceLocation:	0.0
Dcm:SliceThickness:	80.0
Dcm:SoftwareVersion(s):	
	11.1.21.1.2.0Groscan PMS/DICOM 2.0 MR $Id: datadefs,v 5.27 2004/10/18 06:50:Groscan PMS/DICOM 2.0 MR $Id: datadefs,v 5.27 2004/10/18 06:50:Groscan PMS/DICOM 2.0 MR $Id: datadefs,v 5.27 2004/10/18 06:50:
Dcm:SOPClassUID:	1.2.840.10008.5.1.4.1.1.4
Dcm:SOPInstanceUID:	1.3.46.670589.11.0.0.11.4.2.0.5479.5.3792.2006060110160382255
Dcm:SpacingBetweenSlices:	80.0
Dcm:SpecificCharacterSet:	ISO_IR 100
Dcm:StationName:	INTERA
Dcm:StudyDate:	20060601
Dcm:StudyDescription:	TORAX
Dcm:StudyID:	202384105
Dcm:StudyInstanceUID:	1.3.46.670589.11.0.0.11.4.2.0.5479.5.3740.2006060109482582158
Dcm:StudyTime:	094826
Dcm:TemporalPositionIdentifier:	3
Dcm:TransmittingCoil:	B
Signature:	990350f33ac151fac3852555911a9abf6dfe5af6332bdd00f82d7fd8eb7
Tainted:	False

ANNEX B – La Compressió MPEG2

MPEG2 és el nom que rep un conjunt d'estàndards de codificació d'àudio i vídeo que s'utilitza per exemple, en les pel·lícules DVD, en la TDT, etc.
Està pensat per la codificació de imatges en moviment oferint la possibilitat d'associar-hi àudio. El flux de vídeo pot està format per tres tipus de imatges, o frames. Aquests frames són:

- Frames intra (I)
- Frames predictibles (P)
- Frames predictibles bidireccionalment (B)

En el cas dels frames I, la totalitat de la informació és codificada, en canvi els frames P i B prèviament són tractats per un procés de compensació de moviment, que relaciona els frames amb la imatge anterior i els B són relacionats amb l'anterior i la posterior. Cada macrobloc de la imatge P o B s'associa a una àrea de la imatge anterior o següent.

Els frames I codifiquen redundància espaial i els frames B i P redundància temporal. Com que la majoria de vegades els blocs posteriors són iguals, tenim que els quadrats P poden ser un 90% més petits que un I, i els B encara més petits.

Aquests tres tipus de imatges (I, P, B) s'estructuren en un ordre predeterminat en una estructura de 12 o 15 frames específic formant el GOP (Group of Pictures). Hi ha moltes estructures possibles però les més utilitzades tenen una longitud de 12 o 15 frames. I_BB_P_BB_P_BB_P_BB_P_BB_ en seria un exemple. La relació dels quadres dins l'estructura dependrà de la naturalesa del flux i de l'ampla de banda, a més a més el temps de codificació també és un factor a tenir en compte. Un flux que tingui varis frames B pot trigar tres vegades més temps a ser codificat que un flux que només tingui frames I.

La imatge del vídeo es separa en dos parts, luminància Y i croma, que a la vegada es subdivideix en macroblocs que són la unitat bàsica dins d'una imatge. Cada macrobloc es divideix en 8x8 blocs de luminància. Un format comú i que és l'emprat pel Main Profile, és 4:2:0 que és precisament el que s'ha programat que utilitzi el compressor del Starviewer. A la Figura 51 es pot veure quina és la informació que es guardaria per 4 píxels.

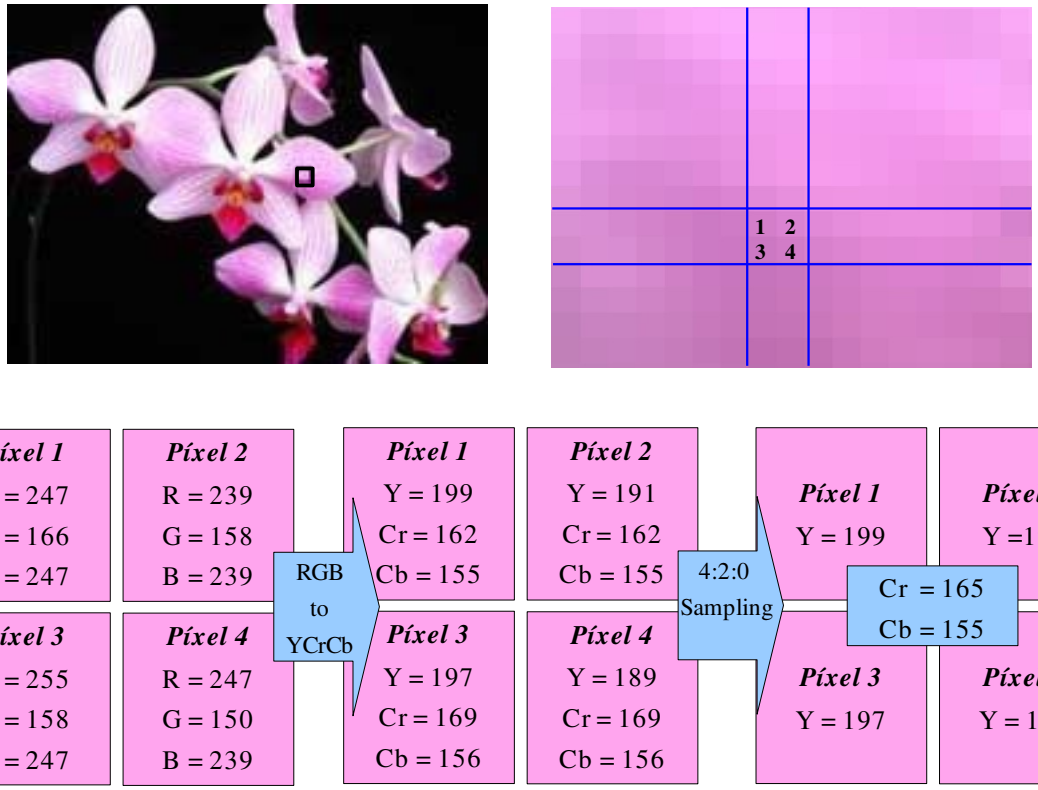


Figura 51: Emmagatzematge del color en un vídeo MPEG2

Bibliografia

[MPEG2] *Free MPEG Software*. www.mpeg.org. Consultada el 29 de juny http://www.mpeg.org/MPEG/MSSG/#source

[COMP212] *Compressor de MPEG2*. www.mpeg.org. Consultada el 29 de juny ftp://ftp.mpegtv.com/pub/mpeg/mssg/mpeg2v12.zip

[VALGRIND] Valgrind. Valgrind Developers. Consultada el 29 de juny http://www.valgrind.org

[GDB] GDB: The GNU ProjectDebbuger. GDB Developers. Consultada 29 de juny http://www.gnu.org/software/gdb/

[UML] Rumbaugh J., Jacobson I., Booch G. (2000). *El lenguaje unificado de modelado. Manual de Referencia.* Madrid. Addison-Wesley

[VTK] Schroeder W., Martin K., Lorensen B. (2002-2004). *The Visualization Toolkit (3rd ed.).* Kitware

[ITK] Ibáñez L., SchroederW. (2003). *The ITK Software Guide.* Kitware